Computational Methods for Partitioned Simulation Coupling:

Applications in Multi-Physics Simulations
and Energy Infrastructure Optimisation

Von dem Stuttgarter Zentrum für Simulationswissenschaften (SC SimTech) und
der Fakultät für Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines Doktors
der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von
Kyle Davis
aus Kapstadt

Hauptberichterin:	Prof. Dr. Miriam Schulte
Mitberichter 1:	Prof. Dr. Rainer Helmig
Mitberichter 2:	Prof. Dr. Michael Bader
Tag der mündlichen Prüfung:	30. Januar 2023

Institut für Parallele und Verteilte Systeme der Universität Stuttgart

2023

Submitted to the University of Stuttgart

Involved institutions and departments:
Cluster of Excellence in Simulation Technology
Institute for Parallel and Distributed Systems
Chair of Simulation of Large Systems

Kyle Davis
Simulation of Large Systems
Institute for Parallel and Distributed Systems
University of Stuttgart
Universitätsstr. 38
70569 Stuttgart
Germany

Contents

Lists of Figures, Tables, Algorithms, and Theorems

List of Figures

List of Tables

List of Algorithms

Abstract/Kurzzusammenfassung

Abstract

Partitioned simulation coupling allows for existing single-physics/single-component software tools to be combined to provide new solutions for more complex interactions. This is often beneficial over monolithic methods, as software specifically designed to solve each single-physics task can be reused. In this thesis, two areas of partitioned simulation coupling are explored: surface coupled multi-physics problems and groundwater heat pump optimisation.

In Part I, numerical methods used to perform partitioned simulation coupling are examined and improved upon. Specifically, the expensive quasi-Newton equation coupling algorithms and radial basis function interpolation are evaluated. In this thesis, new algorithms that reduce the number of QR-decompositions performed during the quasi-Newton update are developed, thereby reducing the simulation runtime. A partition-of-unity method was implemented to perform radial basis function interpolation that is fast, scalable and accurate. Good default parameters were selected based on the numerical testing of both the quasi-Newton and radial basis function algorithms.

In Part II, a new partitioned simulation-optimisation coupling is introduced to optimise the use of shallow groundwater heat pumps throughout a city. Groundwater heat pumps offer a renewable means of heating and cooling buildings in urban environments, but significant interference between the heat pumps can occur, reducing the efficiency and potentially causing large changes to the shallow subsurface water temperatures. In this thesis, a novel coupling approach to combine a subsurface numerical groundwater simulation with an energy infrastructure optimiser is developed. This allows for the optimisation to place heat pumps throughout the city in a cost-optimal manner while accounting for the influence that the new heat pumps have in the subsurface and existing heat pumps. The new partitioned simulation-optimisation solver is applied to a region in the city of Munich to select the optimal locations of heat pumps while remaining within operational constraints. Finally, a novel deep learning surrogate model was developed to help reduce the simulation-optimisation coupling dependence on expensive high-fidelity numerical simulations. The surrogate model is able to capture the thermal influence in the subsurface due to a single heat pump.

Kurzzusammenfassung

Durch die Kopplung von Simulationen in Teilbereichen können bestehende Einzelphysik-/Einzelkomponentensoftwaretools kombiniert werden, um neue Lösungen für komplexere Wechselwirkungen zu finden. Dies ist oft vorteilhafter als monolithische Methoden, da Software, die speziell für die Lösung jeder einzelnen Aufgabe entwickelt wurde, wiederverwendet werden kann. In dieser Arbeit werden zwei Bereiche der partitionierten Simulationskopplung untersucht: oberflächengekoppelte Multiphysikprobleme und Grundwasserwärmepumpenoptimierung.

In Teil I werden die numerischen Methoden für die Kopplung in partitionierten Simulationen untersucht und verbessert, insbesondere die Algorithmen zur Kopplung mittels Quasi-Newton-Methoden und die Interpolation mit radialen Basisfunktionen. In dieser Arbeit werden neue Algorithmen entwickelt, die die Anzahl der während der Quasi-Newton-Aktualisierung durchgeführten QR-Zerlegungen reduzieren und damit die Simulationslaufzeit verkürzen. Eine Partition-of-Unity-Methode wurde implementiert, um eine schnelle, skalierbare und genaue Radialbasisfunktionsinterpolation durchzuführen. Auf der Grundlage der numerischen Tests der Quasi-Newton- und Radialbasisfunktionsalgorithmen wurden gute Defaultparameter ausgewählt.

In Teil II wird eine neue partitionierte Simulations-Optimierungs-Kopplung eingeführt, um den Einsatz von Grundwasserwärmepumpen in einer Stadt zu optimieren. Grundwasser-Wärmepumpen bieten eine nachhaltige Möglichkeit zum Heizen und Kühlen von Gebäuden in städtischen Umgebungen, allerdings es kann zu erheblichen Interferenzen zwischen den Wärmepumpen kommen, die die Effizienz verringern und möglicherweise große Änderungen der Wassertemperaturen im oberflachennahen Untergrund verursachen. In dieser Arbeit wird ein neuartiger Kopplungsansatz entwickelt, um eine numerische Grundwassersimulation mit einem Energieinfrastrukturoptimierer zu kombinieren. Dies ermöglicht die kostenoptimale Platzierung von Wärmepumpen im gesamten Stadtgebiet unter Berücksichtigung des Einflusses, den die neuen Wärmepumpen auf den Untergrund und auf bereits existierende Wärmepumpe haben. Der neue partitionierte Simulations-Optimierungs-Solver wird auf eine Region in der Stadt München angewandt, um die optimalen Standorte für Wärmepumpen auszuwählen und dabei die betrieblichen Einschränkungen zu berücksichtigen. Schließlich wurde ein neuartiges Deep-Learning-Surrogatmodell entwickelt, um die Abhängigkeit der Simulations-Optimierungs-Kopplung von teuren numerischen High-Fidelity-Simulationen zu verringern. Das Surrogatmodell ist in der Lage, den thermischen Einfluss einer einzelnen Wärmepumpe auf den Untergrund zu erfassen.

Acknowledgements

This thesis would not have been possible without the support of others who have all contributed in many different ways.

Firstly, I would like to thank my supervisor Prof. Dr. Miriam Schulte. Thank you for your support, patience and insight throughout this thesis, especially when my focus wandered too far onto other interesting research ideas. I do not think that this work would have been possible without your guidance.

Moving around the world to do a PhD in a new country was difficult, but having a great group of friends can lessen the burden. I would like to thank all of my amazing colleagues at SGS and SC, and Amin Totounferoush for many days of talking and laughing in the office together. To the members of the preCICE team: Amin, Florian, Benni, Makis, Frederic, David, Ishaan and Benjamin, thank you for great time working together and having fun at the coding days together, even though I could not always join the social events. To the Geo.KW team: Fabian, Smajil, Leo, Viktoria and Thilo. Thanks for the many hours working together on something incredibly unique and relevant. I hope that this project grows from what we have started.

Thanks to Team Gelb and the Wellness Climbing groups for all of our wonderful outdoor excursions to the walls and filling my time in Germany with many fond memories. I still have many Grand Tours and, of course, *La Fiamma* to complete.

Of course, thanks goes to all of my friends and family back in SA for the constant support and very flexible schedules when I visited on very short notice. To Maria and Sam, thank you for always believing in me throughout this time. The pandemic did not make it any easier being so far away. Finally, to my fiancé Claudia, who has been a constant source of patience and motivation. You have done so much to make this possible.

Stuttgart, October 27, 2022
Kyle Davis

1 Introduction

It is indisputable that modern computing has impacted every part of our lives. Continuous development in scientific computing, along with improvements in high-performance computing, are having unimaginable impacts in the fields of finance, earth sciences, medicine, astrophysics, telecommunication and engineering to name a few. Simulations of our physical environment provide an extra layer of information alongside theory and experimentation that we might otherwise never be able to access. Where physical experiments are too costly or even impossible to perform, we are able to use computer simulations to fill the gap and provide the missing puzzle pieces.

A challenging simulation field within scientific computing is the field of multi-physics simulation. Multi-physics simulations are characterised as the simultaneous solution of multiple physical components or systems, interacting with each other. As opposed to solving each physical phenomenon independently, the tight interaction between physical fields in multi-physics simulations can capture more intricate physical behaviour, that would otherwise be unobtainable.

In the past, simulations of complex multi-physics interactions seemed out of reach due to their additional computational expense compared to single-physics solvers. Lately, interest has surged due to the improvement of numerical methods and coupling algorithms, parallel computing and performance improvements in computer hardware. Today, we have access to large scale computing clusters and supercomputers around the world. With this enormous computational power at our fingertips, we are constantly striving to solve larger and larger simulation problems. However, with great computational power comes great computational modelling responsibility. Software designed to run on computing clusters should not needlessly waste valuable resources. Therefore, multi-physics simulation solvers should meet multiple requirements; they should be fast, accurate, and scalable to thousands of computing cores, all while remaining as efficient as reasonably possible.

In Section 1.1 of this chapter, the question "what are multi-physics and multi-component simulations" is addressed, followed by an explanation of multi-physics/multi-component simulation coupling methods. In Section 1.2, the specific goals of this thesis are discussed, along with the application fields where the efforts of this work have been applied to. Finally, the contributions of this thesis are presented in Section 1.3, followed by the thesis structure in Section 1.4.

1.1 Welcome to the Multi-Physics-Verse

Multi-physics and multi-component systems are all around us. Multi-physics simulations simultaneously solve two or more individual physical systems, each defined by their own set of physical equations [Key12] and include their mutual interactions into an overall large system. These physical interactions may include various combinations of chemical reactions, electrostatics, heat transfer, structural mechanics, acoustics, and fluid dynamics to name a few. A popular sub-field within multi-physics simulations is fluid-structure interaction (FSI), characterised by the interplay between a fluid and a solid body. In this case, the forces that a fluid exerts on a solid body causes the body to deform, which in turn, influences the magnitude of the forces acting on the solid body by the fluid. The wind blowing through the leaves of a tree is a good example that is simple to understand: the wind blowing over the leaves and branches applies a force on the tree, causing the tree to move/deform, which in return influences the direction and velocity of the wind, and once again, the magnitude and direction of the force from the wind on the tree. This bi-directional interaction between physical fields is not only present in FSI, but in any combination of physical systems. For example, the simulation of a turbine engine could require combining fluid dynamics and solid mechanics to determine the deformation of the turbine blades due to the airflow through the turbine, combined with a heat transfer model to account for the heat dissipation in the turbine blades.

Every day we see countless examples of multi-physics systems interacting in the natural and engineering fields. In aeronautical engineering, multi-physics simulations are used to predict the movement of aircraft wings [Cav07]; [Ris19] or the deformation of wind turbine blades under load [Kor14]; [Hsu12]. Another aeronautical application is that of thin-membrane structures such as parachutes [Sat07] and sails [Par13]. Multi-physics simulations have also been applied in earth sciences, for example within cyclone modelling [Ric19] or hydraulic fracture simulation [Sch22] of subsurface structures. Biomedical engineering has also benefited greatly from multi-physics simulations, where numerous studies have been performed on modelling arterial flow [Paz21]; [Nan21] and FSI of heart valves [Ter20]; [Joh22].

For all the multi-physics examples provided, the question remains how these independent physical models can be combined to provide a realistic, multi-physics solution? There are two fundamental methods to enforce the tight coupling between the physical fields for multi-physics or multi-component problems, namely the monolithic or the partitioned approach.

Monolithic. In a monolithic coupling system, all physical components are solved within a single, global system of equations. This often requires that all components utilise the same discretization and time-stepping schemes. As all individual physical models, as well as their interactions, are built together into a single solver, highly robust and accurate simulations of complex multi-physics problems are often achieved. However, this comes at the cost of flexibility of the monolithic approach as the solver needs to be tailored to the specific application – including numerical aspects such as specialised preconditioners – which cannot easily be modified and must often be built from scratch. Therefore, any modification of the coupling problem comes at a significant expense and extended software development time.

Partitioned. In contrast to the monolithic method, the partitioned coupling method uses independent solvers, e.g., a computational fluid dynamics (CFD) or computational solid mechanics (CSM) solver, where the physical solvers only exchange information across a coupling interface. Each single-physics sub-component can be designed to perform a single role, and to utilise the best discretization and time stepping schemes specific to its own application. Greater flexibility can be achieved for the overall coupling problem, as new sub-components could easily be added into an already existing problem to enhance accuracy or extend the physical modelling capability in a plug-and-play approach. This black-box coupling approach also allows for the combination of an arbitrary number of closed or open-source solvers to be combined in ways that pure commercial or open-source solvers do not support. Partitioned coupling has clear advantages from a software development and flexibility point of view but requires further numerical techniques in order to achieve the robustness and speed of the monolithic approach.

The disadvantage of the partitioned coupling approach is the need for additional numerical schemes to establish strong coupling and data interpolation across the coupling interface. Stability issues and oscillatory behaviour can occur due to the partitioning of equations of the solvers. For example, a well-known issue in FSI occurs when the solid density approaches the density of the fluid material, resulting in the added mass affect. Furthermore, as each solver may use different interface meshes or discretisations, data interpolation is required for black-box coupling based only on point cloud data, i.e., without knowing details about the internal function representation within the solvers. Both of the above problems are compounded by the potential reduction in parallel scaling efficiency, due to both the scalability of the numerical methods required for partitioned coupling, and that of the solvers themselves. All these disadvantages make partitioned coupling a non-trivial challenge for all multi-physics scenarios.

1.2 Goals of this Thesis

The efforts of this thesis have the common goal of improving the state-of-the-art partitioned coupling methods for multi-physics simulations and is split into two different application areas. Part I of the thesis focuses on evaluating and enhancing numerical methods required for partitioned simulation coupling and was a key focus of the *preDOM* project (Section 1.2.1). Part II of this thesis focuses on developing a partitioned simulation coupling framework to perform shallow geothermal energy infrastructure optimisation and formed part of the *GEO.KW* project (Section

1.2.2). Therefore, the aim of this work was not only to develop sophisticated numerical methods to handle partitioned simulation coupling, but also to extend the knowledge of partitioned coupling methods to new application areas.

1.2.1 Part I: preDOM Project

To facilitate partitioned coupling, software tools gluing several single-physics solvers together to establish multi-physics simulation environments have been developed. Of these, preCICE [Cho22] is a popular library of choice, as it is an open-source, general partitioned coupling tool designed to allow for multiple single-physics software codes to be joined together in a minimally invasive manner. Many of the components required for partitioned coupling are implemented in preCICE, such as coupling acceleration, data and time interpolation across interfaces, and parallel data communication. The combination of advanced features and easy to use application programming interface provides an excellent starting point for anyone needing to couple various physics solvers together.

Constant development is required for an open-source software to survive. To improve the general sustainability, usability and performance of preCICE, the *preDOM* project was started. The advanced algorithmic and numerical methods implemented in preCICE ensures a high level of performance and makes it an attractive software when requiring partitioned simulation coupling. However, this comes at the expense of complicated user parameter selection. Numerous user input parameters are required in order to optimally use the advanced functionality. Therefore, every user is faced with the challenge of finding a good set of input parameters for their unique application, greatly influencing the robustness and efficiency of the coupling model.

To bridge the gap between new users and efficient coupling simulations, two aims must be met: *(i)* robust methods that are computationally efficient for a wide range of combinations of input parameters must be implemented, and *(ii)* default parameters that provide "good" coupling performance for most cases are required.

Part I of this thesis focuses on enhancing the performance of two important numerical aspects of preCICE: the coupling acceleration using quasi-Newton acceleration, and data interpolation using radial basis functions (RBF). For these two methods, it is often difficult for users to understand how the parameter selection influences the simulation coupling performance and runtime. The enhancements begin by studying the current quasi-Newton acceleration and RBF interpolation methods implemented in preCICE, developing newer and more efficient methods, and testing them for a variety of applications. Finally, a good, but not necessarily optimal, input parameter set is suggested such that, if no input configuration is given, the default values would be sufficient for setting up a coupled simulation.

1.2.2 Part II: GEO.KW Project

As the European Union aims to increase its usage of renewable energy to 32% by 2030 [Cou18], reducing the energy required for heating and cooling of buildings is an essential step. A shallow **g**round**w**ater **h**eat **p**ump (GWHP) is a highly efficient device that has long been considered an important step towards reducing urban building heating and cooling demands [Sel13]; [Bay12]. Planning of shallow geothermal energy infrastructure is a complicated task due to the mutual interaction of GWHPs, that simultaneously use the subsurface groundwater as both a source and sink of hot and cold water. The *GEO.KW* project was initiated to tackle this challenge by developing a planning and optimisation tool for the efficient thermal utilisation of groundwater as an energy source.

Typical GWHPs operate through a system of extraction and injection wells, where water is extracted from a shallow subsurface aquifer, run through a heat exchanger, and re-injected into the aquifer at a different temperature. This temperature change causes thermal plumes to develop in the aquifer, which may interact with other downstream systems. The mass roll-out of GWHPs in a confined urban environment has the disadvantage that these systems do not operate independently of each other, and significant thermal interactions between GWHPs occur.

Despite knowing about these mutual interactions, in practice, GWHPs are planned and approved individually. Additionally, the seasonal fluctuation of GWHP usage, or potential synergies between various parties, are often not considered. Combining the fact that the approval is mostly performed independently, and that there is a high interaction of GWHP systems, this makes an optimised city-wide layout of GWHP infrastructure highly challenging.

In Part II, we develop a partitioned coupling procedure for a GWHP optimisation and planning tool, that combines a thermal groundwater resource model coupled with an energy demand and infrastructure optimisation solver, to efficiently optimise the GWHP layout and usage on the urban scale of a city. The thermal groundwater model accurately models the thermal impact that all active GWHPs in the optimisation model have on the shallow aquifer. This is coupled to an energy infrastructure optimiser which decides where the optimal locations for new GWHPs are, considering localised energy demands, installation and running costs, natural constraints, and aquifer temperatures. To implement the coupling between solvers, we utilise the coupling schemes, data interpolation and parallel communication, already available in preCICE, to perform a novel coupled simulation for such a planning tool.

1.3 Contributions

The main contributions throughout this thesis build on the knowledge and capabilities of partitioned simulation coupling:

1. We analyse and evaluate various state-of-the-art methods to enhance quasi-Newton methods for partitioned coupling acceleration. We develop new methods to reduce the computa-

tional expense of two stabilisation ingredients of quasi-Newton methods – pre-scaling and filtering. We study the influence that various input parameters have on simulation coupling performance and recommend good default values to improve usability (Chapter 3).

2. We analyse the RBF interpolation method implemented in preCICE and PyRBF[1] for data mapping between non-matching meshes. We develop a highly scalable partition of unity RBF data interpolation solver in PyRBF[2]. We show how this simplifies finding good default values for data interpolation parameters without requiring expensive parameter optimisation, again enhancing usability (Chapter 4).
3. We develop a novel coupling procedure (Chapter 6) for the optimisation of GWHP infrastructure on a city-wide scale. We utilise the newly developed coupling framework to optimise the locations and usage of GWHPs within a sub-region in the city of Munich (Chapter 7), coupling a highly complex numerical groundwater simulation solver and an energy infrastructure optimisation solver.
4. We develop a novel surrogate model using deep learning to determine the thermal field caused by a GWHP. This model is trained on a small number of numerical simulations, while providing accurate results for the local influence of a GWHP (Chapter 8).

1.4 Structure of the Thesis

Part I: Numerical Methods for Partitioned Simulation Coupling.

Chapter 2: Partitioned Multi-Physics Simulation with preCICE – Methods & Software. This chapter introduces the coupling library preCICE and discusses the key partitioned coupling concepts for multi-component simulation coupling with preCICE. Partitioned coupling schemes and equation coupling methods are introduced, followed by an introduction into quasi-Newton methods. Data mapping methods that are available in preCICE, including RBF mapping, are introduced. The limitations of the quasi-Newton and RBF implementation in preCICE are discussed.

Chapter 3: Quasi-Newton Methods for Coupling Acceleration. This chapter provides a description of additional computational enhancements that are required for efficient and robust quasi-Newton coupling acceleration. New computational methods are introduced that reduces the computational overhead of the quasi-Newton implementation in preCICE, and the results from numerous partitioned problems are presented.

Chapter 4: Data Interpolation for Simulation Coupling. Common RBF interpolation difficulties are discussed, and methods to reduce the computational cost are introduced. We present a fast and scalable RBF solver developed in this work, that is able to simplify the difficult parameter

[1]https://github.com/floli/PyRBF
[2]https://github.com/KyleDavisSA/PyRBF

selection for data interpolation while simultaneously reducing the computational cost of RBF interpolation.

Part II: Geothermal Energy Infrastructure Optimisation.

Chapter 5: Shallow Geothermal Resource Optimisation. This chapter introduces the problem of shallow geothermal energy usage optimisation from the GEO.KW project, followed by a description of the role of simulation coupling for shallow groundwater heat pump optimisation. The individual software components required for simulation coupling are introduced, namely the numerical groundwater simulation solver and the energy infrastructure optimisation solver.

Chapter 6: Coupling Schemes for Partitioned Shallow Geothermal Resources Optimisation. The newly developed simulation-optimisation coupling concept is explained, which forms the main contribution of Part II of this thesis. A simple example is presented to explain the flow of information and constraint checking.

Chapter 7: Analysis of Coupling Schemes for Geothermal Energy Optimisation. The results from two coupled numerical simulation-optimisation test cases are presented in this chapter. The first test case validates the developed coupling concept on a relatively simple but realistic model. The second test case presents results from a real-world test case from the city of Munich.

Chapter 8: Deep Learning for Shallow Subsurface Modelling. A novel surrogate model using deep learning to provide fast evaluations of GWHP thermal plumes is presented. Various components in the training pipeline, including neural network design, data generation, and network training are presented. Finally, training results for the deep learning model and recommendations for future development are discussed.

Chapter 9: Conclusion. A summary of this thesis and its main contributions are presented. We highlight the successful completion of various goals of the thesis and discuss the current limitations of this work. Finally, future research topics are proposed.

Part I

Numerical Methods for Partitioned Simulation Coupling

2 Partitioned Multi-Physics Simulation with preCICE – Methods & Software

Partitioned coupling schemes are attractive options when performing multi-physics and multi-component simulations, as existing software codes and solvers can be leveraged for faster development times and allow higher flexibility in the range of physical systems that can be added. To efficiently perform partitioned coupling, a few key components are required: *(i)* communication between solvers that could each be running in parallel on distributed systems *(ii)* data exchange between solvers across non-matching interfaces, requiring interpolation between the surfaces (or volumetric domains in the case of volumetric coupling), and *(iii)* efficient iterative equation coupling to enable fast convergence of the coupled problem.

In this chapter, we begin by introducing the coupling library *preCICE*, that is the software basis for this work, and discuss its capabilities in section 2.1. Next, partitioned coupling schemes for time-dependent problems are introduced in section 2.2, followed by a short introduction to the quasi-Newton method used for fast iterative coupling in section 2.2.1. Additional pre-scaling and filtering techniques, which improve the stability and convergence rate of the quasi-Newton methods for equation coupling, are presented in sections 2.2.3 and 2.2.4. Section 2.3 introduces data interpolation methods between non-matching coupling interfaces. Radial basis function interpolation is introduced in section 2.3.2, where their advantages, disadvantages and current implementation in preCICE are discussed. In both sections 2.2.5 and 2.3.5, the difficulty of preparing the input parameter configurations for preCICE to perform optimal coupling acceleration and interpolation, thereby hindering the usability of preCICE, is discussed.

2.1 Introduction to preCICE

As multi-physics simulations offer significant advantages compared to individual physics simulations only, numerous multi-physics solvers and coupling libraries have been developed. The coupling library preCICE was developed by Bernhard Gatzhammer [Gat15], based on a previous library called FSI*ce, initially developed by Markus Brenk [Bre07]. The focus of preCICE's development was to be minimally invasive, offer all required components for black-box simulation coupling, and remain highly scalable on high performance computing (HPC) systems. The principal components of preCICE are shown in Figure 2.1, including: communication, data mapping, coupling schemes and time interpolation. Each physics solver, being any one of *CFD solver*, *FEM solver*, *in-house solver* or *particle solver*, is modified by introducing a solver adapter into the source code for open-source projects, or into an application program interface (API) for closed-source software. This enables the solver to communicate with preCICE through function calls (*libprecice*) via the preCICE API. This ease of integrating preCICE into a new solver makes it an attractive option for simulation coupling. However, preCICE is not the only coupling software available.

Similar alternative coupling softwares are MpCCI [Wol17], DTK [Sla13], and OpenPALM [Duc15]. MpCCI[1] is a commercial FSI coupling software developed by the Fraunhofer Institute for Algorithms and Scientific Computing. MpCCI offers ready-to-use adapters for a variety of commercial and open-source codes, as well as an API to be integrated into further codes. DTK[2] is an open-source coupling library developed at the Oak Ridge National Laboratory. The application programming interface (API) offers lower-level features compared to preCICE, allowing more flexibility regarding the coupling logic, but at a greater development effort for the user. OpenPALM[3], developed by CERFACS and ONERA, is also an open-source coupling software that offers a higher-level approach, with built-in coupling logic and a graphical user interface. As the work of this thesis formed part of the preDOM project, we specifically focus on the development of preCICE and do not perform an in-depth analysis of alternative coupling software here. However, preCICE combines various of key features that are found, in part, across a variety of alternatives. Detailed comparisons of partitioned coupling libraries, including others not mentioned here, are available in [Cho22].

As preCICE has no understanding of the physics equations in the solver itself, almost any simulation software can be coupled to preCICE. To describe how preCICE functions, each section of Figure 2.1 is briefly explained: *(i)* Communication – each coupled solver is a separate executable that could be running on multiple computing nodes. Once preCICE is called from each computing rank, all asynchronous communication is handled by preCICE only between computing ranks that need to exchange data and is based on either MPI Ports or TCP/IP. *(ii)* Data mapping – interpolation of coupled variables between non-matching, decomposed coupling interfaces are handled through either projection-based or radial basis function interpolation (see Section 2.3). *(iii)* Coupling

[1]https://www.mpcci.de/
[2]https://github.com/ORNL-CEES/DataTransferKit
[3]https://cerfacs.fr/globc/PALM_WEB/index.html

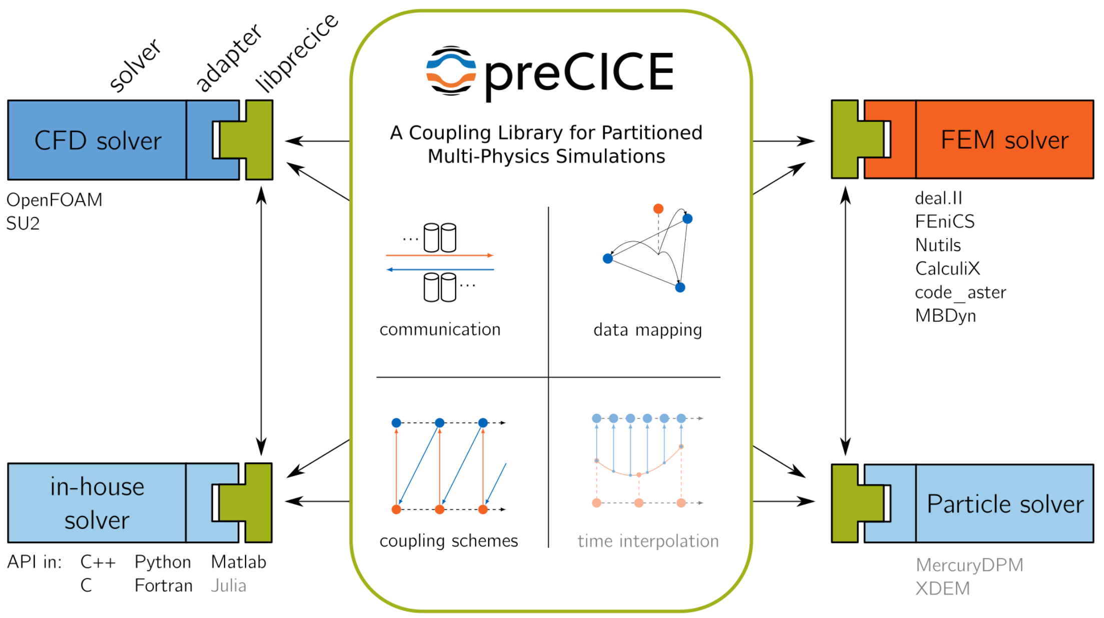


FIGURE 2.1 Building blocks and key capabilities of the preCICE coupling library. The coupled simulation software directly calls preCICE. Therefore, solver adapters (*adapters*) that connect preCICE to the solvers are embedded in the solver itself, which could be an open-source, closed-source or in-house solver, and contain function calls to the preCICE library (*libprecice*). All coupled solvers are able to communicate with all other solvers, and have access to all of preCICE's main functionality: parallel communication, data mapping, coupling schemes, acceleration schemes and time integration. Image available at `https://github.com/precice/precice.github.io/tree/master/material` .

Schemes – defines the logical coupling flow of information between solvers, controlling which participants exchange data and when. Coupling acceleration (see Section 2.2) is included in the coupling scheme logic. *(iv)* Time interpolation – higher-order interpolation is still under development, however, sub-cycling is supported.

In the rest of the chapter, the *(ii)* acceleration and *(iii)* data interpolation functionality, as currently implemented in preCICE, are explained in more detail.

2.2 Partitioned Coupling Schemes for Time-Dependent Problems

For time-dependent multi-physics problems, partitioned coupling can be divided into two types: explicitly (loosely) coupled or implicitly (strongly) coupled. For explicit coupling, each solver performs each time step once[4], data are exchanged across the coupling interface, and the solver proceeds to the next time step. For implicit coupling, all solvers perform each time step

[4]We do not concern ourselves here with the internal functioning of the solver and treat it purely as a black-box. One coupling iteration is complete when the time step computation ends.

repeatedly, exchanging data at the end of each repetition, which is called a coupling iteration. For simplicity, we consider only two coupled solvers when deriving the fixed-point equations that are solved in each coupling iteration. The two solvers are represented by functions $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ operating on data $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ on the coupling interface $\boldsymbol{\Gamma}$. However, the method generalises to any number of coupled solvers.

A common example of a multi-physics simulation is a fluid-structure interaction problem, where the fluid solver is represented by the operator $\boldsymbol{S}_1$ that maps the coupling interface displacements or velocities $\boldsymbol{x}_1$ to forces $\boldsymbol{x}_2$ exerted on the structure. The structural solver, operator $\boldsymbol{S}_2$, maps the coupling interface forces $\boldsymbol{x}_2$ to interface displacements or velocities $\boldsymbol{x}_1$. The mapping $\boldsymbol{S}_1$ requires the output of $\boldsymbol{S}_2$ and vice-versa such that

$$\boldsymbol{S}_1 : \boldsymbol{x}_1 \mapsto \boldsymbol{x}_2 \quad \text{and} \quad \boldsymbol{S}_2 : \boldsymbol{x}_2 \mapsto \boldsymbol{x}_1. \tag{2.1}$$

For the time-dependent problems, the variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are the respective interface values at the new time step. Strong/implicit coupling between $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ can be formulated as a fixed-point problem, where two mathematically equivalent variants of this fixed-point problem exist. The first variant results in a Gauss-Seidel type system:

$$\boldsymbol{x}_1 = \boldsymbol{S}_2 \circ \boldsymbol{S}_1(\boldsymbol{x}_1), \quad \text{with} \quad \boldsymbol{x}_2 = \boldsymbol{S}_1(\boldsymbol{x}_1), \tag{2.2}$$

and the Jacobi type system:

$$\begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & \boldsymbol{S}_2 \\ \boldsymbol{S}_1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix}, \tag{2.3}$$

in matrix-like notation. Both Gauss-Seidel and Jacobi type coupling methods can be condensed into a single formulation

$$\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{x}, \quad \text{where} \quad \boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{S}_2 \circ \boldsymbol{S}_1(\boldsymbol{x}_1) \quad \text{or} \quad \boldsymbol{H}(\boldsymbol{x}) = \begin{pmatrix} 0 & \boldsymbol{S}_2 \\ \boldsymbol{S}_1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix}, \tag{2.4}$$

and the residual

$$\boldsymbol{R}(\boldsymbol{x}) := \boldsymbol{H}(\boldsymbol{x}) - \boldsymbol{x} = 0. \tag{2.5}$$

The simplest method for solving the fixed-point problem is to use the unmodified fixed-point iterations $\boldsymbol{x}^{k+1} = \boldsymbol{H}(\boldsymbol{x}^k)$. Explicit coupling in its simplest form corresponds to a single fixed-point iteration, starting with an initial guess $\boldsymbol{x}^0$ that usually equals the solution of the previous time step at the coupling interface, and ends with the solution $\boldsymbol{x}^1$ without any check for convergence. Accordingly, explicit coupling can be either Gauss-Seidel type (each solver executes their time step one after the other), or Jacobi type (both solvers execute their time step in parallel). Implicit

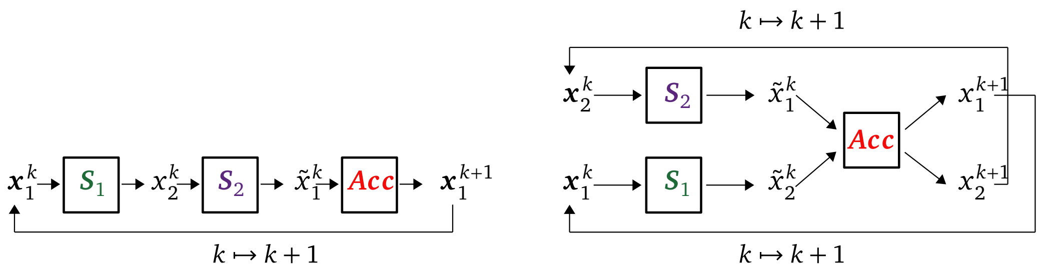


FIGURE 2.2 Schematic of serial (Gauss-Seidel) and parallel (Jacobi) type implicit coupling schemes for partitioned coupling of two solvers $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$. An acceleration scheme $\boldsymbol{Acc}$ modifies the fixed-point iterate $\boldsymbol{x}_1^k$ and $\boldsymbol{x}_2^k$ through the strategy $\boldsymbol{x}^k \to \tilde{\boldsymbol{x}}^k \to \boldsymbol{x}^{k+1}$. Quasi-Newton techniques (Sec. 2.2.1) can be used to accelerate the fixed-point iteration.

coupling solves multiple fixed-point iterations successively, where the initial guess $\boldsymbol{x}^k$ is the output from the previous coupling iteration, until a convergence threshold is achieved. However, this may be slow to converge, or completely diverge all together for strongly coupled problems. Additionally, the evaluation of the operator $\boldsymbol{H}$ is typically very expensive as it requires the execution of a time step in each single solver. To enhance the robustness, and to reduce the number of evaluations of $\boldsymbol{H}$, an additional acceleration step $\boldsymbol{Acc}()$, i.e., a suitable post-processing of the iteration result, is required:

$$\boldsymbol{x}^{k+1} = \boldsymbol{Acc}(\tilde{\boldsymbol{x}}^k) \ \text{ with } \ \tilde{\boldsymbol{x}}^k = \boldsymbol{H}(\boldsymbol{x}^k).$$

The implicit coupling schemes with the additional acceleration step are depicted in Figure 2.2. Gauss-Seidel type coupling typically suffers from poor parallel performance on large computing clusters due to the serial execution of the solvers in $\boldsymbol{H}(\boldsymbol{x})$, which leads to idling computing resources. The Jacobi type coupling improves the parallel performance as all solvers run simultaneously, but the concatenation of coupled surface variables $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)^T$ adds additional complexity due to the magnitude difference of the field variables, which is addressed in Section 2.2.3.

Convergence. The implicit partitioned coupling approach requires a convergence criterion/threshold to determine when the fixed-point problem is sufficiently solved. If the convergence criterion is built only using the operator $\boldsymbol{R}(\boldsymbol{x})$, then convergence is evaluated in terms of a single variable only for Gauss-Seidel type coupling, whereas all variables are evaluated for Jacobi type coupling ($\boldsymbol{x}_1$ and $\boldsymbol{x}_2$). To perform generalised partitioned coupling for any coupled test case, all coupling variables should be considered, as this is the closest criterion to achieving the solution of the monolithic formulation [Uek16]. As the interface variables $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)^T$ could live on different scales, a relative residual is used as the convergence criterion for stopping the coupling iterations within one time step, i.e., to trigger moving to the next time step. The convergence criterion based on the relative residual is defined as

$$\frac{\|\tilde{\boldsymbol{x}}_i^k - \boldsymbol{x}_i^k\|_2}{\|\tilde{\boldsymbol{x}}_i^k\|_2} = \frac{\|\boldsymbol{R}_i(\boldsymbol{x}_i)\|}{\|\tilde{\boldsymbol{x}}_i^k\|_2} < \epsilon_{\text{conv}} \tag{2.6}$$

for $i = 1, 2$ for Gauss-Seidel type coupling and $i = 1, 2, \ldots$ for Jacobi type coupling[5]. The residual operator $\boldsymbol{R}_i$ indicates the component of the residual $\boldsymbol{R}$ for $\boldsymbol{x}_i$ only. The convergence threshold ϵ_{conv} is a user defined parameter. The convergence criterion of the various solvers needs to be smaller/stricter than the coupling convergence ϵ_{conv}.

Fixed-point acceleration. Simply using the unmodified fixed-point iterations $\boldsymbol{x}^{k+1} = \boldsymbol{H}(\boldsymbol{x}^k)$ is often not sufficient to achieve convergence. Common acceleration schemes used for multi-physics simulations are either under-relaxation methods or quasi-Newton methods. The under-relaxed fixed-point iteration is defined as

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \omega^k(\tilde{\boldsymbol{x}}^k - \boldsymbol{x}^k) \tag{2.7}$$

with the under-relaxation parameter $\omega^k \in (0, 1]$. The simplest version is constant under-relaxation, i.e., $\omega^k = \omega \quad \forall k$. This is simple to implement, but often not sufficient for robustness or convergence speed. An improved dynamic under-relaxation method, also referred to as Aitken relaxation [Iro69]; [Küt08], is defined as

$$\omega^{k+1} = -\omega^k \frac{(\boldsymbol{R}^{k-1})^T(\boldsymbol{R}^k - \boldsymbol{R}^{k-1})}{\|\boldsymbol{R}^k - \boldsymbol{R}^{k-1}\|_2^2},$$

with the residual $\boldsymbol{R}^k = \tilde{\boldsymbol{x}}^k - \boldsymbol{x}^k$. Aitken relaxation methods typically outperform constant under-relaxation methods. However, quasi-Newton acceleration methods have been shown to be far superior [Deg09]; [Bog16b]; [Uek16]; [Sch18]. In the following, we introduce quasi-Newton methods in general, including the specific variants and enhancements that are already implemented in preCICE at the beginning of this thesis' work.

2.2.1 Introduction to Quasi-Newton Methods

Multi-secant quasi-Newton methods have become popular methods for applications where only black-box solver information is available and classical inexact Newton methods are not feasible. Initially developed in a different community in the context of acceleration of fixed-point solvers under the name Anderson mixing or Anderson acceleration [And65], quasi-Newton methods now find application in a wide variety of areas under different names. Over the years, multiple formulations and variations have been developed, with mathematically equivalent formulations having been derived independently by Oosterlee and Washio [Oos00], Miller [Mil05], and Degroote et. al. [Deg09]. Especially relevant to the multi-physics community is the least-squares formulation

[5]Jacobi type coupling easily extends to more than 2 sub-fields, allowing for $i = 1, 2, \ldots$.

by Degroote et. al. [Deg09], the block-iterative method (IBQN-LS) by Vierendeels et. al. [Vie07] and the multi-vector (MV) method by Bogaers et. al. [Bog14].

Newton methods are standard methods for solving nonlinear root-finding problems such as our fixed-point problem $\boldsymbol{R}(\boldsymbol{x}) := \boldsymbol{H}(\boldsymbol{x}) - \boldsymbol{x}$. However, there are multiple reasons why full Newton methods are often undesirable for problem sizes typically encountered in multi-physics simulations. Fang and Saad [Fan09] provide a list of reasons why so-called multi-secant quasi-Newton methods are beneficial over full Newton methods. Firstly, the derivatives $\nabla\boldsymbol{R}(\boldsymbol{x})^T$ are too expensive to compute or are inaccessible for black-box coupling. Secondly, even if the derivatives are available, the dimension of $\boldsymbol{R}^N$ is large and often results in excessive memory usage, where N is the total number of degrees of freedom (DoF). Lastly, the cost of evaluating $\boldsymbol{H}(\boldsymbol{x})$ is high and may contain noise. To avoid these issues, quasi-Newton methods approximate the system Jacobian's inverse $\nabla\boldsymbol{R}(\boldsymbol{x})^{-T}$ using secant information.

Secant information corresponds to approximating a single derivative by a difference quotient but requires less input-output pairs than entries in the Jacobian. In addition, they store the evaluations of $\boldsymbol{H}(\boldsymbol{x})$ by using information gathered throughout previous coupling iterations. According to Taylor's theorem, the approximate inverse Jacobian $\boldsymbol{J}^{-1} \approx (\nabla\boldsymbol{R})^{-T}$ must satisfy the secant equation $\Delta\boldsymbol{x}_i^k = \boldsymbol{J}^{-1}\Delta\boldsymbol{r}_i^k$, where the input and output information are defined as $\Delta\boldsymbol{x}_i^k = \boldsymbol{x}^k - \boldsymbol{x}^i$ and $\Delta\boldsymbol{r}_i^k = \boldsymbol{r}^k - \boldsymbol{r}^i$ with $\boldsymbol{r}^i = \boldsymbol{R}(\boldsymbol{x}^i)$ and $\boldsymbol{x}^i$ close to the current iteration $\boldsymbol{x}^k$.

Instead of a single secant-equation, more information can be used to formulate a multi-secant equation

$$\overline{\boldsymbol{J}}^{-1}\boldsymbol{V}_k^\eta = \overline{\boldsymbol{W}}_k^\eta, \tag{2.8}$$

with the input and output difference pairs collected in the matrices $\boldsymbol{V}_k^\eta$ and $\overline{\boldsymbol{W}}_k^\eta$ given as

$$\boldsymbol{V}_k^\eta = \left[\Delta\boldsymbol{r}^k, \Delta\boldsymbol{r}^{k-1}, \dots, \Delta\boldsymbol{r}^{k-\eta}\right] \quad \text{with} \quad \Delta\boldsymbol{r}^i = \boldsymbol{r}^i - \boldsymbol{r}^{i-1}, \tag{2.9}$$

$$\overline{\boldsymbol{W}}_k^\eta = \left[\Delta\boldsymbol{x}^k, \Delta\boldsymbol{x}^{k-1}, \dots, \Delta\boldsymbol{x}^{k-\eta}\right] \quad \text{with} \quad \Delta\boldsymbol{x}^i = \boldsymbol{x}^i - \boldsymbol{x}^{i-1} \tag{2.10}$$

from previous iterations, where $\boldsymbol{V}_k^\eta \in \boldsymbol{R}^{N\times\eta}$, $\overline{\boldsymbol{W}}_k^\eta \in \boldsymbol{R}^{N\times\eta}$ and $\overline{\boldsymbol{J}}^{-1} \in \boldsymbol{R}^{N\times N}$ in the k^{th} iteration, and the number of columns η. For many applications, the assumption is that $\eta \ll N$ holds. Therefore, Equation 2.8 is a highly under-determined system for $\overline{\boldsymbol{J}}^{-1}$. In order to find a unique solution, further norm-minimisation needs to be performed. Therefore, the inverse Jacobian is determined by searching for the minimiser

$$||\overline{\boldsymbol{J}}^{-1}||_F \to \min, \tag{2.11}$$

to obtain a unique solution for the approximation $\overline{\boldsymbol{J}}^{-1}$. The solution at the next iteration $\boldsymbol{x}^{k+1}$ is determined by

$$\boldsymbol{x}^{k+1} = \tilde{\boldsymbol{x}}^k + \overline{\boldsymbol{J}}^{-1}\boldsymbol{R}(\boldsymbol{x}^k), \tag{2.12}$$

In this formulation of updating $\boldsymbol{x}^k$ to $\boldsymbol{x}^{k+1}$, the dimensionality of the space spanned by columns of $\overline{\boldsymbol{W}}_k$ does not increase after the first two iterations. This leads to linearly dependent updates and causes the solution to stagnate. Therefore, the update step for the quasi-Newton method is reformulated using $\tilde{\boldsymbol{x}}^k$ and $\boldsymbol{J}^{-1}$ according to [Deg09] as

$$\boldsymbol{x}^{k+1} = \tilde{\boldsymbol{x}}^k + \boldsymbol{J}^{-1}\tilde{\boldsymbol{R}}(\boldsymbol{x}^k), \tag{2.13}$$

with the modified fixed-point operator

$$\widetilde{\boldsymbol{R}}(\tilde{\boldsymbol{x}}) := \tilde{\boldsymbol{x}} - \boldsymbol{H}^{-1}(\tilde{\boldsymbol{x}}). \tag{2.14}$$

Matrix $\widetilde{\boldsymbol{V}}_k = \boldsymbol{V}_k$ remains unchanged as $\widetilde{\boldsymbol{R}}(\tilde{\boldsymbol{x}}^k) = \boldsymbol{R}(\boldsymbol{x}^k)$, whereas $\overline{\boldsymbol{W}}$ is modified to include the formulation in terms of $\tilde{\boldsymbol{x}}$

$$\boldsymbol{W}_k^{\eta} = \left[\Delta\tilde{\boldsymbol{x}}^k, \Delta\tilde{\boldsymbol{x}}^{k-1}, \ldots, \Delta\tilde{\boldsymbol{x}}^{k-\eta}\right] \quad \text{, with} \quad \Delta\tilde{\boldsymbol{x}}^i = \tilde{\boldsymbol{x}}^i - \tilde{\boldsymbol{x}}^{i-1} \tag{2.15}$$

Summarising, the multi-secant quasi-Newton method results in a very efficient update step as we do not have to use a linear solver within each Newton step to determine the inverse Jacobian, as we directly approximate it. In addition, only input and output difference values of the operator evaluations $\boldsymbol{H}(\boldsymbol{x})$ are used, which were already performed in previous iterations.

The quality of the inverse Jacobian approximation is dependent on the quality and quantity of information stored in $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ (the superscript η is dropped for brevity). The computational complexity and memory requirements also grow with the size of N and η. To increase the amount of information in $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$, we can define a number of previous time steps, ζ, from which information is included in $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ in addition to information from iterations in the current time step. Therefore, the total number of columns, η, is dependent on ζ. In the following, we omit the superscript for brevity and only refer to $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$.

An additional measure to include already collected information from the past in our inverse Jacobian approximation is to generalise the norm minimisation in Equation 2.11 to

$$\min \|\boldsymbol{J}^{-1} - \boldsymbol{J}_{prev}^{-1}\|_F, \tag{2.16}$$

which contains information older than ζ time steps in $\boldsymbol{J}_{prev}^{-1}$, and was first introduced in [Bog14] and further expanded in [Sch18]. A generic update formula of the norm minimisation, including the multi-secant information and derived in Uekermann [Uek16] and Scheufele [Sch18] is

$$\boldsymbol{J}^{-1} - \boldsymbol{J}_{prev}^{-1} = (\boldsymbol{W}_k - \boldsymbol{J}_{prev}^{-1}\boldsymbol{V}_k)(\boldsymbol{V}_k^T\boldsymbol{V}_k)^{-1}\boldsymbol{V}_k^T = \widetilde{\boldsymbol{W}}_k\boldsymbol{V}_k^{\dagger}, \tag{2.17}$$

where $\boldsymbol{V}_k^{\dagger} = (\boldsymbol{V}_k^T\boldsymbol{V}_k)^{-1}\boldsymbol{V}_k^T$ is the pseudo-inverse of $\boldsymbol{V}_k$ and $\widetilde{\boldsymbol{W}}_k = (\boldsymbol{W}_k - \boldsymbol{J}_{prev}^{-1}\boldsymbol{V}_k)$.

2.2.2 Quasi-Newton Variants

In the section below, we derive the most popular quasi-Newton variations that are often applied to multi-physics simulations and are implemented in preCICE. For further in-depth analysis and comparison of multi-secant quasi-Newton methods, the reader is referred to Uekermann [Uek16].

IQN-ILS

The Interface Quasi-Newton Inverse Least-Squares (IQN-ILS) method is a popular and frequently used multi-physics coupling acceleration scheme. It was first introduced in [Deg09], and is similar to the Anderson acceleration formulation by Donald G. Anderson to accelerate fixed-point iterations [And65]. For the *IQN-ILS* method, we set $\boldsymbol{J}_{prev}^{-1} = 0$, and therefore solve

$$\boldsymbol{J}^{-1}\boldsymbol{V}_k = \boldsymbol{W}_k \quad \text{with} \quad \boldsymbol{J}^{-1} = \operatorname{argmin}||\boldsymbol{J}^{-1}||. \tag{2.18}$$

Inserting $\boldsymbol{J}_{prev}^{-1} = 0$ into Equation 2.17 yields

$$\boldsymbol{J}^{-1} = \boldsymbol{W}_k \left(\boldsymbol{V}_k^T \boldsymbol{V}_k\right)^{-1} \boldsymbol{V}_k^T = \boldsymbol{W}_k \boldsymbol{V}_k^{\dagger}, \tag{2.19}$$

where $\boldsymbol{V}_k^{\dagger} = (\boldsymbol{V}_k^T \boldsymbol{V}_k)^{-1} \boldsymbol{V}_k^T$ is the pseudo-inverse computation of $\boldsymbol{V}_k$. As no previous inverse Jacobian $\boldsymbol{J}_{prev}^{-1}$ information is stored, the performance (robustness and convergence rate) of the *IQN-ILS* method is completely dependent on the quality of information stored in $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$, which is dependent on the total number of columns η stored over the previous ζ time steps. As the total number of columns increases, $\boldsymbol{V}_k$ may become rank deficient due to the accumulation of linearly dependent columns. This requires so-called filtering to recover a well-conditioned problem by removing columns as further explained in Section 2.2.4.

A big advantage of the *IQN-ILS* approach is the option for a matrix-free implementation ($\boldsymbol{J}^{-1}$ is not stored in memory) of the quasi-Newton update step by directly computing the update step as

$$x^{k+1} = \tilde{\boldsymbol{x}}^k + \boldsymbol{W}_k \boldsymbol{\alpha} \;\text{ with }\; \boldsymbol{\alpha} = \operatorname{argmin}||\boldsymbol{V}_k \boldsymbol{\alpha} + \boldsymbol{r}^k||_2. \tag{2.20}$$

The coefficient vector $\boldsymbol{\alpha} = \boldsymbol{V}_k^{\dagger} \boldsymbol{r}^k$ is calculated by computing a QR-decomposition $\boldsymbol{V}_k = \boldsymbol{QR}$ and solving $\boldsymbol{R\alpha} = -\boldsymbol{Q}^T \boldsymbol{r}^k$.

IQN-IMVJ

The Interface Quasi-Newton Inverse Multi-Vector Jacobian method [Bog14]; [Sch18]; [Spe20] implicitly retains information from previous time steps in $\boldsymbol{J}_{prev}^{-1}$. The inverse Jacobian $\boldsymbol{J}^{-1}$ stays as close as possible to $\boldsymbol{J}_{prev}^{-1}$ through norm minimisation from Equation 2.16

$$||\boldsymbol{J}^{-1} - \boldsymbol{J}_{prev}^{-1}|| \to \min,$$

while incorporating the multi-secant information shown in Equation 2.17. Therefore, $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ only need to store information gathered after $\boldsymbol{J}_{prev}^{-1}$. For time-dependent multi-physics applications, a natural starting point is to update the previous approximation $\boldsymbol{J}_{prev}^{-1}$ after each time step, and therefore $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ only need to retain the columns from the current time steps. This reduces the potential for $\boldsymbol{V}_k$ to become rank-deficient, as well as reducing the cost of matrix vector computations. Once again, the pseudo-inverse $\boldsymbol{V}_k^{\dagger}$ is computed using a QR-decomposition.

The disadvantage of this method is that we need an explicit representation of $\boldsymbol{J}_{prev}^{-1}$, which requires $\mathcal{O}(N^2)$ both in memory and computational complexity, making the *IQN-IMVJ* method unsuitable for large problems of N. The other disadvantage is that the *IQN-IMVJ* method does not have the ability to easily remove old information that is potentially outdated or conflicting with the current behaviour of the coupled problem. Once potentially conflicting information is stored in $\boldsymbol{J}_{prev}^{-1}$, it is difficult to remove. However, its impact decreases over the time steps due to the continuous updates of $\boldsymbol{J}_{prev}^{-1}$.

Scheufele [Sch18] showed that by smart approximations, both the computational and memory cost of $\boldsymbol{J}_{prev}^{-1}$ can be reduced to $\mathcal{O}(N)$. Re-visiting equation 2.17 and re-writing by moving $\boldsymbol{J}_{prev}^{-1}$ to the right-hand side we find that,

$$\boldsymbol{J}^{-1} = \boldsymbol{J}_{prev}^{-1} + (\boldsymbol{W}_k - \boldsymbol{J}_{prev}^{-1}\boldsymbol{V}_k)(\boldsymbol{V}_k^T\boldsymbol{V}_k)^{-1}\boldsymbol{V}_k^T = \boldsymbol{J}_{prev}^{-1} + \widetilde{\boldsymbol{W}}_k\boldsymbol{V}_k^{\dagger}. \tag{2.21}$$

Unrolling the successive summations of $\boldsymbol{J}^{-1} = \boldsymbol{J}_{prev}^{-1} + \widetilde{\boldsymbol{W}}_{k_n}\boldsymbol{V}_{k_n}^{\dagger}$ for the n^{th} time step, $\boldsymbol{J}_{prev}^{-1}$ can be approximated by

$$\boldsymbol{J}_{prev}^{-1} = \widetilde{\boldsymbol{W}}_{k_0}\boldsymbol{V}_{k_0}^{\dagger} + \widetilde{\boldsymbol{W}}_{k_1}\boldsymbol{V}_{k_1}^{\dagger} + \widetilde{\boldsymbol{W}}_{k_2}\boldsymbol{V}_{k_2}^{\dagger} + \ldots + \widetilde{\boldsymbol{W}}_{k_{n-1}}\boldsymbol{V}_{k_{n-1}}^{\dagger}. \tag{2.22}$$

By default, preCICE stores all iterations from a time step i into a single group $\widetilde{\boldsymbol{W}}_{k_i}$ and $\boldsymbol{V}_{k_i}^{\dagger}$. The Newton update step can be computed on-the-fly by matrix-vector computations without explicitly building $\boldsymbol{J}_{prev}^{-1}$ by

$$\Delta\boldsymbol{x}^{k+1} = \tilde{\boldsymbol{x}}^k - \sum_{q=0}^{n}\widetilde{\boldsymbol{W}}_{k_q}(\boldsymbol{V}_{k_q}^{\dagger}\boldsymbol{r}^k). \tag{2.23}$$

This simple computation allows for the *IQN-IMVJ* method to be used for multi-physics simulations even for large values of N. As $\widetilde{\boldsymbol{W}}_{k_n} \neq \boldsymbol{W}_{k_n}$, the new column $\boldsymbol{w}_k = \Delta\tilde{\boldsymbol{x}}_{k-1}^k$ cannot simply be added to $\widetilde{\boldsymbol{W}}_{k_n}$. The inverse Jacobian approximation is simply the sum of matrices $\widetilde{\boldsymbol{W}}_{k_n}$ and $\boldsymbol{V}_{k_n}^{\dagger}$ from previously completed time steps, and thus $\boldsymbol{J}_{prev}^{-1}$ remains constant within one time step. The new column for $\widetilde{\boldsymbol{W}}_{k_n}$ is defined as

$$\widetilde{\boldsymbol{w}}_k = \Delta\tilde{\boldsymbol{x}}^k - \boldsymbol{J}_{prev}^{-1}\boldsymbol{r}^k = \Delta\tilde{\boldsymbol{x}}^k - \sum_{q=0}^{n-1}\widetilde{\boldsymbol{W}}_{k_q}(\boldsymbol{V}_{k_q}^{\dagger}\boldsymbol{r}^k), \tag{2.24}$$

which follows from $\widetilde{\boldsymbol{W}}_{k_n} = (\boldsymbol{W}_{k_n} - \boldsymbol{J}_{prev}^{-1}\boldsymbol{V}_{k_n})$. The sum in Equation 2.23 may become infeasibly long after many time steps. Therefore, Scheufele [Sch18] introduces a restart, i.e., re-setting $\boldsymbol{J}_{prev}^{-1}$ to a cheap representation after a "chunk" of m time steps.

Restart Algorithms for J_{prev}^{-1}:

The purpose of the restart algorithms is to condense the information within the Jacobian approximation into a reduced form, reducing the memory storage requirements of $\widetilde{\boldsymbol{W}}_k$ and $\boldsymbol{V}_k^\dagger$. The information from m previous time steps is condensed into two new smaller matrices $\widetilde{\boldsymbol{W}}_{k_{res}}$ and $\boldsymbol{V}_{k_{res}}^\dagger$, such that

$$\widetilde{\boldsymbol{W}}_{k_{res}} \boldsymbol{V}_{k_{res}}^\dagger \approx \sum_{q=n-m}^{n} \widetilde{\boldsymbol{W}}_{k_q} \boldsymbol{V}_{k_q}^\dagger, \tag{2.25}$$

where the size of the new $\widetilde{\boldsymbol{W}}_{k_{res}}$ and $\boldsymbol{V}_{k_{res}}^\dagger$ is much smaller than the sum of the sizes of $\widetilde{\boldsymbol{W}}_{k_q}$ and $\boldsymbol{V}_{k_q}^\dagger$. The simplest option is to drop $\widetilde{\boldsymbol{W}}_{k_q}$ and $\boldsymbol{V}_{k_q}^\dagger$ and clear all information from previous time steps, i.e., $\widetilde{\boldsymbol{W}}_{k_{res}} = \boldsymbol{V}_{k_{res}} = 0$ and, in addition, also $\boldsymbol{W}_k$ and $\boldsymbol{V}_k$ for the current time step are reset to empty matrices. This has the potential benefit of forgetting old or conflicting information, but often results in reduced performance as too much information is lost. Two additional methods developed in [Sch18] are the *RS-LS* and the *RS-SVD* methods.

The RS-LS method clears all $\widetilde{\boldsymbol{W}}_k$ and $\boldsymbol{V}_k^\dagger$, but explicitly retains the secant information from the previous η_m iterations from within the current chunk of m time steps. The new $\boldsymbol{J}_{prev}^{-1}$ is simply determined using

$$\widetilde{\boldsymbol{W}}_{k_{res}} = \boldsymbol{W}_{\eta_m} \quad \text{and} \quad \boldsymbol{V}_{k_{res}} = \boldsymbol{V}_{\eta_m}. \tag{2.26}$$

A more complex method that can approximate the information from all previous time steps is the RS-SVD method, where the dominant modes from a truncated singular value decomposition (SVD) are retained,

$$\boldsymbol{J}_{prev}^{-1} = \widetilde{\boldsymbol{\Psi}}\widetilde{\boldsymbol{\Sigma}}\widetilde{\boldsymbol{\Phi}}^T, \qquad \widetilde{\boldsymbol{W}}_{k_{res}} = \widetilde{\boldsymbol{\Psi}}, \qquad \boldsymbol{V}_{k_{res}}^\dagger = \widetilde{\boldsymbol{\Sigma}}\widetilde{\boldsymbol{\Phi}}^T, \tag{2.27}$$

where $\widetilde{\boldsymbol{\Psi}},\widetilde{\boldsymbol{\Phi}} \in R^{N\times\kappa}$, $\widetilde{\boldsymbol{\Sigma}} = diag(\sigma_1,\sigma_2,\ldots,\sigma_\kappa) \in R^{\kappa\times\kappa}$, and κ is the number of modes from a truncated SVD. The inverse Jacobian is approximated by $\boldsymbol{J}_{prev}^{-1} = \widetilde{\boldsymbol{\Psi}}\widetilde{\boldsymbol{\Sigma}}\widetilde{\boldsymbol{\Phi}}^T$. The approximation is obtained after truncating all singular values of the full SVD below a user specified threshold ϵ_{svd}. The SVD approximation only holds an advantage over the RS-LS method if the SVD computation is fast, and if the inverse Jacobian can be well approximated by a low rank approximation. As computing a complete SVD is computationally intensive, a SVD update procedure was implemented in preCICE by [Sch18] using an efficient, low rank SVD-update method, following the work of [Bra06]. The update method is able to add new columns of information to an already computed SVD. In more detail, the decomposition of the sum of length m starts with an SVD of the first summands $\widetilde{\boldsymbol{W}}_{k_0}\boldsymbol{V}_{k_0}^\dagger$. All other summands from $\widetilde{\boldsymbol{W}}_{k_1}\boldsymbol{V}_{k_1}^\dagger$ to $\widetilde{\boldsymbol{W}}k_m\boldsymbol{V}_{k_m}^\dagger$ are added to $\widetilde{\boldsymbol{W}}_{k_0}\boldsymbol{V}_{k_0}^\dagger$ using the SVD update procedure, where all modes smaller than ϵ_{svd} are truncated after each SVD update step. This ensures that $\widetilde{\boldsymbol{W}}_{k_{res}} \boldsymbol{V}^{\dagger}_{k_{res}}$ remain small.

The above-mentioned quasi-Newton variants exhibit different characteristics in terms of computational cost and memory storage. However, two general techniques are useful to improve the convergence rate and robustness for both *IQN-ILS* and *IQN-IMVJ* methods: pre-scaling and filtering.

2.2.3 Pre-scaling

When performing coupled multi-physics simulations, the coupling variables between different solvers may live on different scales. For fluid-structure interactions, for example, the fluid forces may be orders of magnitude larger than the solid displacements or velocities. For Gauss-Seidel type coupling, this does not impact quasi-Newton coupling as only a single set of coupling data is used for the acceleration step. However, for Jacobi type coupling, the fixed-point acceleration occurs on the vector $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)^T$[6] , where all coupling data from all solvers are concatenated into a single vector. Not only could these values live on different scales, the same can hold for their residuals $\boldsymbol{R}(\boldsymbol{x}_k)$. This may cause numerical deficiencies as only the field of larger magnitude is "seen" in the inverse approximation $\boldsymbol{J}^{-1}$. An example application is the flow around a rigid solid structure, where the interface pressure may be in the order of 10^5, while the structural deformation may be on the order of 10^{-2}. Likewise, their residuals may be equally different. To tackle this problem, pre-scaling is performed by replacing $\boldsymbol{V}_k$ and $\boldsymbol{r}^k$ in Equation 2.20 and Equation 2.23 by

$$\boldsymbol{V}_k^{'} = \boldsymbol{\Lambda}_k \boldsymbol{V}_k \quad \text{and} \quad \boldsymbol{r}^{k^{'}} = \boldsymbol{\Lambda}_k \boldsymbol{r}^k,$$

where $\boldsymbol{\Lambda}_k = \text{diag}\left(\lambda_{k,1} \dots \lambda_{k,N}\ \lambda_{k,N+1} \dots \lambda_{k,2N}\right)^T$ and N is the number of DoF in each solver[7]. Uekermann [Uek16] explored the use of a *per-entry* pre-scaling, where each data value in $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)^T$ and $\boldsymbol{R}(\boldsymbol{x}) = (\boldsymbol{R}_1(\boldsymbol{x}_1), \boldsymbol{R}_2(\boldsymbol{x}_2))^T$ is scaled with a unique scaling value $\lambda_{k,i}$, and a *per-sub-vector* pre-scaling, where all entries in each sub-vector $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and $\boldsymbol{R}_1(\boldsymbol{x}_1)$ and $\boldsymbol{R}_2(\boldsymbol{x}_2)$ are scaled by a single sub-vector scaling $\lambda_{k,i} = \bar{\lambda}_{k,1}$ for $i = 1, \dots, N$, and $\lambda_{k,i} = \bar{\lambda}_{k,2}$ for $i = N+1, \dots, 2N$. The two *per-sub-vector* pre-scaling values are concatenated as $\bar{\boldsymbol{\Lambda}}_k = (\bar{\lambda}_{k,1}, \bar{\lambda}_{k,2})^T$. Uekermann [Uek16] found that the *per-entry* pre-scaling does not offer any benefit over the *per-sub-vector* method. Therefore, we only consider the *per-sub-vector* variant.

There are various pre-scaling options available in preCICE. However, the recommended option is the *residual-sum* pre-scaling, initially introduced by [Mar08], but modified to sum the values over all iterations in one time step:

[6]In this example we consider two fields for simplicity, but the pre-scaling method generalises to an arbitrary number of fields $\boldsymbol{x} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_i)^T$

[7]The acceleration step is performed for all exchanged variables on a single coupling interface. Therefore, the pre-scaling is performed after the data interpolation step such that all information is on the same mesh

$$\bar{\lambda}_{k,1} = \sum_{j=1}^{k} \frac{||\boldsymbol{S}_2(\boldsymbol{x}_2^j) - \boldsymbol{x}_1^j||_2}{||\boldsymbol{R}(x^j)||_2} \quad \text{and} \quad \bar{\lambda}_{k,2} = \sum_{j=1}^{k} \frac{||\boldsymbol{S}_1(\boldsymbol{x}_1^j) - \boldsymbol{x}_2^j||_2}{||\boldsymbol{R}(x^j)||_2}. \tag{2.28}$$

Uekermann [Uek16] showed that the summation over all iterations within one time step alleviates a zig-zag convergence behaviour. At the beginning of each time step, the summation over all previous iterations is set to zero[8]. A detailed description for pre-scaling in the *IQN-IMVJ* method is provided in [Sch18].

Other pre-scaling options include *constant* pre-scaling, which sets $\lambda_{k,i}$ equal to a user specified value, and *value* pre-scaling, which performs the scaling in Equation 2.28 with the data magnitude $\boldsymbol{S}(\boldsymbol{x}^j)$. *Constant* pre-scaling adds an extra user parameter that can impact robustness and convergence performance. The *value* pre-scaling offers a parameter free option but is less suited as we use the pre-scaling values to scale the residuals in $\boldsymbol{V}_k$ and $\boldsymbol{R}(\boldsymbol{x}^k)$.

Pre-scaling adds an additional computational step to the coupling procedure. The QR-decomposition, which is one of the most expensive operations within the quasi-Newton update step, is forced to be recomputed in each iteration as we have to decompose $\boldsymbol{V}_k^{'} = \boldsymbol{\Lambda}_k \boldsymbol{V}_k$. A caveat when using the *IQN-IMVJ* with RS-SVD restart, is that the pre-scaling weights must be frozen during the first SVD construction, as we cannot re-scale $\boldsymbol{V}_k$ in hindsight after having approximated a SVD of $\sum_{q=0}^{n} \widetilde{\boldsymbol{W}}_{k_q} \boldsymbol{V}_{k_q}^{\dagger}$. This requires all matrices in further SVD-update steps to be equally scaled. This potentially limits the use of the RS-SVD method for problems where the underlying physical behaviour changes.

2.2.4 Filtering

Both quasi-Newton variants *IQN-ILS* and *IQN-IMVJ* rely on a QR-decomposition of $\boldsymbol{V}_k$[9]. However, as $\boldsymbol{V}_k$ becomes larger, there is no guarantee that all columns in $\boldsymbol{V}_k$ remain linearly independent. This can cause $\boldsymbol{V}_k$ to become almost singular, rendering the QR-decomposition costly and unstable. The causes of almost linearly dependent columns building up may be due to convergence to a stationary solution for the transient problems, rounding errors of the solver, or too many columns being retained in $\boldsymbol{V}_k$. To alleviate this problem and improve the conditioning of $\boldsymbol{V}_k$, *filtering* of the columns of $\boldsymbol{V}_k$ was introduced [Hae16]. The work of Haelterman et. al. [Hae16] proposed three filtering methods: QR1, QR2 and POD. The POD method, which built on a proper orthogonal decomposition of $\boldsymbol{V}_k$, was shown to offer no benefit compared to the much simpler QR1 and QR2 filters. We therefore focus on these variants only.

A QR-decomposition transforms a matrix into an unitary matrix, and an upper triangular matrix, i.e., $\boldsymbol{V}_k^{\eta} = \boldsymbol{QR}$, where $\boldsymbol{Q} \in \mathbb{R}^{N \times \eta}$ and $\boldsymbol{R} \in \mathbb{R}^{\eta \times \eta}$. Before discussing each filtering variant,

[8]The pre-scaling weights cannot be reset to 1, as this would artificially scale all sub-vectors around 1. This would impact the pre-scaling performance if both sub-vector residuals were orders of magnitude below 1.

[9]We drop the superscript of $\boldsymbol{V}_k^{'}$ for sake of brevity. Before a QR-decomposition is performed of $\boldsymbol{V}_k$, it is first scaled according to $\boldsymbol{V}_k^{'} = \boldsymbol{\Lambda}_k \boldsymbol{V}_k$ if pre-scaling is selected.

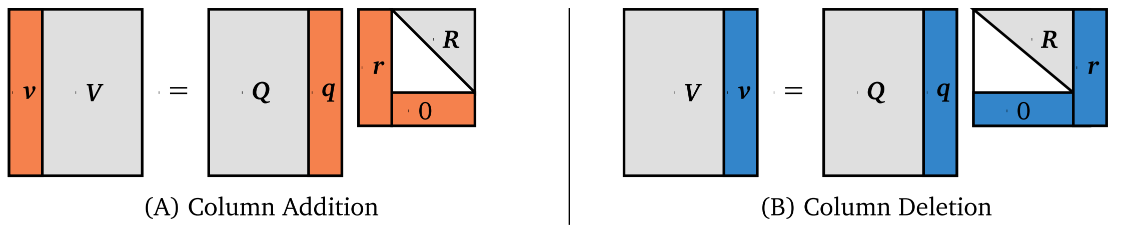


FIGURE 2.3 Schematic of the column insertion and column deletion technique for the QR update. For every new column added to $\boldsymbol{V}_k$ on the left, the QR decomposition can be updated by adding the orthonormalised version of this column as a new (rightmost) column $\boldsymbol{q}$ in $\boldsymbol{Q}$, adding an additional row of zeros to $\boldsymbol{R}$ and subsequently adding a new (leftmost) column $\boldsymbol{r}$ to $\boldsymbol{R}$ representing the respective orthonormalisation factors. Givens rotations are applied to eliminate sub-diagonal entries in $\boldsymbol{R}$. To remove a column from the right of $\boldsymbol{V}_k$, the rightmost column $\boldsymbol{q}$ in $\boldsymbol{Q}$ is removed, and the rightmost column $\boldsymbol{r}$ and bottom row is removed from $\boldsymbol{R}$. This step does not require Givens rotations to remove sub-diagonal entries.

it is necessary to understand how the QR-decomposition is performed in preCICE. A fast column insertion/deletion procedure was implemented in preCICE [Uek16]; [Sch18]: if a new column is added into $\boldsymbol{V}_k$, a new column is added into $\boldsymbol{Q}$ and $\boldsymbol{R}$, without needing to recompute the entire QR-decomposition. By using this *QR-update* procedure, the QR-decomposition is effectively built from right to left of $\boldsymbol{V}_k$ (from the oldest to the newest information in $\boldsymbol{V}_k$). We explain this in more detail in the following paragraph before presenting the basics of the two filtering methods.

QR—Update: Starting from an already computed QR decomposition, $\boldsymbol{Q}$ and $\boldsymbol{R}$ from the previous iteration, a new column is added to the left of $(\boldsymbol{V}_k)_{(:,1)}$[10]. This new column is orthonormalised against $\boldsymbol{Q}$ via a modified Gram–Schmidt procedure. The new orthonormalised column, $\boldsymbol{q}$, is added as an additional (rightmost) column in $\boldsymbol{Q}$. A new column, $\boldsymbol{r}$, is added to the left of $\boldsymbol{R}$, along with a bottom row of zeros. A series of Givens rotations eliminate any non-zero sub-diagonal entries in $\boldsymbol{R}$ to retain the upper triangular structure. The matrices during the column addition and column removal process are depicted in Figure 2.3. A detailed explanation of the update procedure can be found in [Dan76]. Note that this seems to be unnecessarily complicated compared to adding columns to the right of $\boldsymbol{V}_k$, which would only require orthonormalisation of the new column $\boldsymbol{q}$ with a standard Gram–Schmidt algorithm, and adding a new column $\boldsymbol{r}$ on the right of $\boldsymbol{R}$. However, the current method has the benefit that deleting old columns from the right of $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ is cheap, and requires only removing the rightmost columns of $\boldsymbol{Q}$ and $\boldsymbol{R}$, and the bottom rows of $\boldsymbol{R}$. Removing an arbitrary column, j, from the middle of $\boldsymbol{V}_{k,j}$ requires a few more steps: *(i)* removing the corresponding column from $\boldsymbol{R}_{(:,j)}$, *(ii)* removing any sub-diagonal elements from $\boldsymbol{R}$ using Givens rotations, *(iii)* applying the corresponding Givens rotations to $\boldsymbol{Q}$, and *(iv)* removing the last column from $\boldsymbol{Q}$ as well as the bottom (zero) row from $\boldsymbol{R}$. If a complete QR decomposition

[10]the subscript $(:,1)$ indicates the row and column number in Python-like notation. This refers to the left most column of $\boldsymbol{V}_k$.

```
$R_{11} = \|(V_k)_{:,1}\|_2$                                  ⇝ This is the newest column
for $i = 1, ..., \eta$ do                                     ⇝ Starts from newest column
    $\overline{v} = (V_k)_{:,i}$
    for $j = 1, ..., i-1$ do
        $R_{ji} = Q^T_{:,j} \cdot \overline{v}$
        $\overline{v} = \overline{v} - R_{ji} \cdot Q_{:,j}$
    if $\|\overline{v}\|_2 < \epsilon_f \|(V_k)_{:,i}\|_2$ then
        delete column $i$
    $R_{ii} = \|\overline{v}\|_2$ and $Q_{:,i} = \overline{v}/R_{ii}$
```

ALGORITHM 2.1 Pseudo-code for the QR2 Filter [Hae16]. Each column from $\boldsymbol{V}_k$ is added one at a time to $\boldsymbol{Q}$ and $\boldsymbol{R}$ through a column insertion process. This rebuilds the QR-decomposition from the newest column, deleting older, linearly dependent columns during the QR construction process.

is required, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are discarded completely, and the new $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices are rebuilt using each column from $\boldsymbol{V}_k$, adding one column at a time.

QR1: The QR1 filtering step is performed after the QR-update procedure. In the QR1 filter, the degree of linear dependency of a new column compared to previous columns is estimated by comparing the diagonal elements of $\boldsymbol{R}$ to the complete norm of $\boldsymbol{R}$, a metric for the norm of the orthogonalised columns of $\boldsymbol{V}_k$ before normalisation. A column i is deleted if

$$\boldsymbol{R}_{ii} < \epsilon_f \cdot \|\boldsymbol{R}\|_F, \tag{2.29}$$

where ϵ_f is the filtering limit, a user specified parameter. This filter has a potential drawback as the QR-decomposition is built from the oldest columns of $\boldsymbol{V}_k$, and the new column may be removed by the filter of large matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. However, Uekermann [Uek16] found that the QR1 filter performed well with a suitable filter limit. The main advantage of the QR1 filter is its simplicity as it only requires column insertion and deletion steps.

QR2: The QR2 filter was introduced in [Hae16] as a means to quantify the amount of new information a column in $\boldsymbol{V}_k$ adds to the QR-decomposition. It uses the norm of a column after orthogonalisation compared to the norm before orthogonalisation as a refinement criterion, i.e., uses a relative criterion instead of the absolute criterion of the QR1 filter. This filters columns during the construction of $\boldsymbol{Q}$ and $\boldsymbol{R}$ itself, beginning with the newest columns of $\boldsymbol{V}_k$ (leftmost column). Therefore, the QR2 filter completely reconstructs $\boldsymbol{Q}$ and $\boldsymbol{R}$ in each coupling iteration. The QR2 filter only checks if a column is to be deleted when it is added to $\boldsymbol{Q}$ and $\boldsymbol{R}$, and if it is not deleted, it is not reconsidered for deletion again. Therefore, the QR2 filter tends to delete older columns. This contrasts with the QR1 filter, where all columns are considered for deletion. Deleting older columns is favoured is they may no longer be meaningful for the current dynamics of the physical system. The pseudo-code of the QR2 filter is shown in Algorithm 2.1.

Within the multi-physics and the fixed-point acceleration community, filtering has been widely applied to the QR-decomposition step. For the least-squares update, [Mar08] uses a regularisation term $min \ \|\boldsymbol{V}_k\boldsymbol{\alpha} + \boldsymbol{R}^k\|_2^2 + \frac{\beta}{2}\|\boldsymbol{\alpha}\|_2^2, \beta > 0$. Fand and Saad [Fan09] used a POD variation but without deleting columns entirely. Walker and Ni [Wal11] simply deleted the oldest columns if the condition of $\boldsymbol{R}$ dropped below a threshold. Examining alternative methods for filtering, it is clear that there is no method perfectly suited for multi-physics simulations or provably better than other methods. Important considerations for any filtering method are that it must be easy to use with limited user defined parameters and be computationally cheap.

To complete to derivation of the current quasi-Newton implementation in preCICE, the pseudo-code of the *IQN-ILS* and *IQN-IMVJ* methods with additional pre-scaling and filtering steps are shown in Algorithm 2.2.

```
initial value $\boldsymbol{x}^0$
$\tilde{\boldsymbol{x}}^0 = H(\boldsymbol{x}^0)$ and $\boldsymbol{r}^0 = \tilde{\boldsymbol{x}}^0 - \boldsymbol{x}^0$
$\boldsymbol{x}^{k+1} = \boldsymbol{x}^0 + \omega(\tilde{\boldsymbol{x}}^0 - \boldsymbol{x}^0)$
for $k = 1, 2, \ldots$ do
  $\tilde{\boldsymbol{x}}^k = \boldsymbol{H}(\boldsymbol{x}^k)$ and $\boldsymbol{r}^k = \tilde{\boldsymbol{x}}^k - \boldsymbol{x}^k$
  if converged then
    break
  $\Delta\boldsymbol{r}^k = \boldsymbol{r}^k - \boldsymbol{r}^{k-1}$
  $\Delta\tilde{\boldsymbol{x}}^k = \tilde{\boldsymbol{x}}^k - \tilde{\boldsymbol{x}}^{k-1}$
  $\boldsymbol{V}_k = [\Delta\boldsymbol{r}^k, \ldots, \Delta\boldsymbol{r}^1]$
  $\boldsymbol{W}_k = [\Delta\tilde{\boldsymbol{x}}^k, \ldots, \Delta\tilde{\boldsymbol{x}}^1]$
  pre-scaling weights $\boldsymbol{\Lambda}_k$
  Compute $\boldsymbol{\Lambda}_k\boldsymbol{V}_k = \boldsymbol{QR}$
  Filter columns in $\boldsymbol{QR}$
  solve $\boldsymbol{R\alpha} = -\boldsymbol{Q}^T\boldsymbol{\Lambda}_k\boldsymbol{r}^k$
  $\boldsymbol{x}^{k+1} = \tilde{\boldsymbol{x}}^k + \boldsymbol{W}_k\boldsymbol{\alpha}$
```

```
initial value $\boldsymbol{x}^0$
$\tilde{\boldsymbol{x}}^0 = \boldsymbol{H}(\boldsymbol{x}^0)$ and $\boldsymbol{r}^0 = \tilde{\boldsymbol{x}}^0 - \boldsymbol{x}^0$
$\boldsymbol{x}^{k+1} = \boldsymbol{x}^0 + \omega(\tilde{\boldsymbol{x}}^0 - \boldsymbol{x}^0)$
for $k = 1, 2, \ldots$ do
  $\tilde{\boldsymbol{x}}^k = \boldsymbol{H}(\boldsymbol{x}^k)$ and $\boldsymbol{r}^k = \tilde{\boldsymbol{x}}^k - \boldsymbol{x}^k$
  if converged then
    break
  $\Delta\boldsymbol{r}^k = \boldsymbol{r}^k - \boldsymbol{r}^{k-1}$
  $\Delta\tilde{\boldsymbol{x}}^k = \tilde{\boldsymbol{x}}^k - \tilde{\boldsymbol{x}}^{k-1}$
  $\boldsymbol{V}_k = [\Delta\boldsymbol{r}^k, \ldots, \Delta\boldsymbol{r}^1]$
  $\boldsymbol{W}_k = [\Delta\tilde{\boldsymbol{x}}^k, \ldots, \Delta\tilde{\boldsymbol{x}}^1]$
  Pre-scaling weights $\boldsymbol{\Lambda}_k$
  Compute $\boldsymbol{\Lambda}_k\boldsymbol{V}_k = \boldsymbol{QR}$
  Filter columns in $\boldsymbol{QR}$
  Solver $\boldsymbol{RV}_k^\dagger = \boldsymbol{Q}^T$
  $\widetilde{\boldsymbol{W}}_{k_n} = \boldsymbol{W}_{k_n} - \sum_{q=0}^{n-1}\widetilde{\boldsymbol{W}}_{k_q}(\boldsymbol{V}_{k_q}^\dagger\boldsymbol{V}_{k_n})$.
  $\Delta\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \sum_{q=0}^{n}\widetilde{\boldsymbol{W}}_{k_q}(\boldsymbol{V}_{k_q}^\dagger\boldsymbol{r}^k)$
  $\boldsymbol{x}^{k+1} = \tilde{\boldsymbol{x}}^k + \Delta\boldsymbol{x}^{k+1}$
```

ALGORITHM 2.2 Pseudo code for the *IQN-ILS* (left) and *IQN-IMVJ* (right). Both methods require an initial input x^0. Pre-scaling is applied to the matrix $\boldsymbol{V}_k$ before performing a QR-decomposition, followed by filtering of $\boldsymbol{Q}$ and $\boldsymbol{R}$.

2.2.5 Good Practice & Limitations

There already exists good practice methods for configuring input parameters for partitioned multi-physics coupling acceleration with preCICE. Despite the advanced implementation of the described quasi-Newton methods in preCICE, a thorough understanding of the methods, and how they influence the simulation coupling, is required to optimally configure the simulation setup. In this section, we highlight how changing input parameters affects the robustness of the acceleration methods, the memory footprint and the computational cost of the additional pre-scaling, filtering

```
<acceleration:IQN-IMVJ always-build-jacobian="0">
      <initial-relaxation value="{float}" enforce="0"/>
      <imvj-restart-mode truncation-threshold="0.0001" chunk-size="8"
                         reused-time-windows-at-restart="8" type="RS-SVD"/>
      <max-used-iterations value="{integer}"/>
      <time-windows-reused value="{integer}"/>
      <data scaling="1" mesh="{string}" name="{string}"/>
      <filter limit="1e-16" type="{string}"/>
      <preconditioner freeze-after="-1" type="{string}"/>
</acceleration:IQN-IMVJ>
```

FIGURE 2.4 Input parameters with default input values for the *IQN-IMVJ* quasi-Newton coupling acceleration method from the preCICE XML reference. All parameters required for *IQN-ILS* acceleration are also required for the *IQN-IMVJ* acceleration. The reference includes all necessary configurable input parameters along with their data type if no default is provided. The XML reference is available at `https://precice.org/configuration-xml-reference.html`.

and restart algorithms. A typical input parameter set for the *IQN-IMVJ* quasi-Newton method is shown in Figure 2.4.

The parameter *initial-relaxation* defines the value of ω^k in equation 2.7 and is required only for the first coupling iteration when $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ are empty. The default value is 0.1. The *data* variable defines which interface data is used to build the quasi-Newton approximation and is typically all variables exchanged along the coupling interface such as forces and displacements. However, additional sets of data can be exchanged between solvers that are not used for the quasi-Newton acceleration. Configurable parameters that greatly influence the coupling performance are described in Table 2.1.

IQN-ILS

1. As the *IQN-ILS* method builds the approximation of $\boldsymbol{J}^{-1}$ exclusively from $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$, the quality of these matrices is important. Storing more columns in $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ typically improves the convergence rate of the acceleration method. The total number of time steps reused can be set to control how large $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ grow. The preCICE configurable parameters are **max-iterations**, η, and **max-timesteps-reused**, ζ. A choice between $\eta = 100$ and $\eta = 200$ and between $\zeta = 10$ and $\zeta = 20$ is typically advised.
2. Storing more columns in $\boldsymbol{V}_k$ increases the QR-decomposition cost, which grows with $\mathcal{O}(N\eta^2)$.
3. Using the pre-scaling is always recommended for Jacobi/parallel coupled scenarios, as each sub-vector may live on different scales. This comes with the caveat that a complete QR-decomposition is performed in each iteration instead of a cheaper column addition.
4. The QR2 filter is preferred to the QR1 filter, as it builds the QR-decomposition beginning with the latest information. If combined with the pre-scaling, no additional computational cost is

TABLE 2.1 User specified parameters for the *IQN-ILS* and *IQN-IMVJ* method. The input parameter name, variable and description of each parameter is provided. Two additional parameters, **convergence limit** and **max coupling iterations**, that are not present in Figure 2.4 are described. The **max coupling iterations** defines the number of iterations performed before moving onto the next time step, even if convergence is not achieved. All input parameters are shown in Figure 2.4, and a complete description of the input parameters can be found at `https://precice.org/configuration-xml-reference.html`.

Input Parameter	**Variable**	**Description**
Time Steps Reused	ζ	Total number of previous time steps reused in $\boldsymbol{V}_k$.
Max iterations	η	Total number of columns allowed in $\boldsymbol{V}_k$.
Preconditioner	—	Choice of pre-scaling: recommended *residual-sum*
Filter Type	—	Choice of filtering: recommended *QR2*
Filter Limit	ϵ_{filter}	Threshold for deleting columns from $\boldsymbol{V}_k$
SVD Threshold	ϵ_{svd}	Threshold for truncating columns in $\boldsymbol{\Sigma}$
Time Steps at Restart	$\zeta_{restart}$	Number of columns reused for RS-LS restart
Convergence limit	ϵ_{conv}	Convergence threshold for implicit coupling
Max coupling iterations	η_{max}	Maximum number of coupling iterations allowed

incurred with the QR2 filter. A good filter limit is typically between $\epsilon_f = 0.1$ and $\epsilon_f = 0.001$.

IQN-IMVJ

1. By default, preCICE starts with empty $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ at the start of each time step, i.e., **max-timesteps-reused** $= 0$ is recommended. Pre-scaling and QR2 filtering do not increase the computational cost significantly.
2. Storing the entire inverse Jacobian $\boldsymbol{J}_{prev}^{-1}$ is far too expensive for most applications and is not recommended.
3. Choosing a suitable restart algorithm depends on the multi-physics problem.
   a) The RS-SVD method is able to store a reduced representation of all columns from $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ across all time steps, but a suitable threshold, ϵ_{svd}, is required to control the size of $\tilde{\boldsymbol{\Phi}}$, $\tilde{\boldsymbol{\Sigma}}$ and $\tilde{\boldsymbol{\Psi}}$. The pre-scaling weights must also be frozen before the first SVD computation is performed, meaning that the RS-SVD method cannot easily adjust to changing behaviour of the solver residuals. A suitable threshold is typically between $\epsilon_{svd} = 10^{-2}$ and $\epsilon_{svd} = 10^{-4}$.
   b) The RS-LS method can be viewed as a combination of *IQN-ILS* and *IQN-IMVJ*. The number of **timestep-reused-at-restart** specifies how many time steps of information is retained in the restart. All older columns are removed. A suitable value of previous time steps reused is between 10 and 20 time steps.

Despite the advantages that pre-scaling, filtering and *IQN-IMVJ* restarting offer, they introduce additional complication and confusion for new users when selecting input parameters, and add a significant amount of additional computational steps. In chapter 3, we introduce additional enhancements to these methods, with the aim to reduce their sensitivity to the input parameters, and to reduce the computational cost involved.

2.3 Data Interpolation for Information Transfer

Individual physics solvers in a partitioned coupled simulation rarely have matching meshes at the coupling interface. This may be due to restrictions in the meshing software used, the discretization method of the physical solvers, or perhaps the level of mesh refinement necessary to capture specific behaviour of the solver at the interface. As data at the coupling interface are exchanged in each iteration, fast and accurate means of performing a data mapping from one interface mesh to another is critical. An example of a non-matching interface is shown in Figure 2.5. As detailed discretization information, such as mesh connectivity and underlying finite element basis functions, may not be available in black-box partitioned coupling, interpolation methods that can function on point cloud data are required.

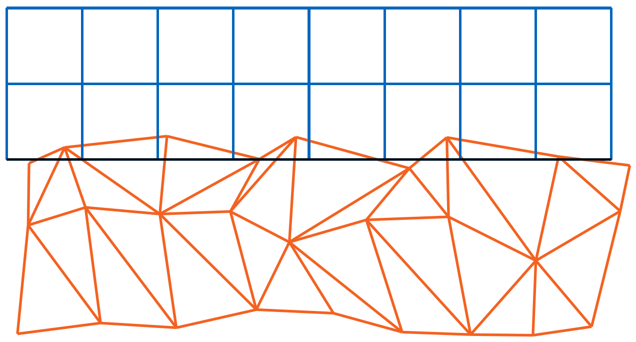


FIGURE 2.5 Example of two 2D surface meshes with a non-matching coupling interface. Values at the vertices (defined as the intersection point of edges) of the structured grid (blue) are mapped to the vertices of the unstructured grid (orange) at the common interface (black). Data mapping requires that information at vertices that lie on the common interface are exchanged between the two meshes.

Using similar terminology from Section 2.2, consider a scenario of two participants, $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, that share a common geometrical interface. We define sets of points $\boldsymbol{\Gamma}_1$ and $\boldsymbol{\Gamma}_2$ discretising the interface in $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, respectively as

$$\boldsymbol{\Gamma}_1 = \{\boldsymbol{x}_1^{\Gamma_1}, \dots, \boldsymbol{x}_{N_{\Gamma_1}}^{\Gamma_1}\} \quad \text{with} \quad \boldsymbol{x}_i^{\Gamma_1} \in \mathbb{R}^d \quad \text{and} \quad \boldsymbol{\Gamma}_2 = \{\boldsymbol{x}_1^{\Gamma_2}, \dots, \boldsymbol{x}_{N_{\Gamma_2}}^{\Gamma_2}\} \quad \text{with} \quad \boldsymbol{x}_i^{\Gamma_2} \in \mathbb{R}^d. \tag{2.30}$$

The data mapping procedures aims to map the data $\boldsymbol{v}^{\Gamma_1} = (v_1^{\Gamma_1}, v_2^{\Gamma_1}, \dots, v_{N_{\Gamma_1}}^{\Gamma_1})^T$ at the vertices of $\boldsymbol{\Gamma}_1$ to the vertices of $\boldsymbol{\Gamma}_2$, i.e., determine the values $\boldsymbol{v}^{\Gamma_2} = (v_1^{\Gamma_2}, v_2^{\Gamma_2}, \dots, v_{N_{\Gamma_2}}^{\Gamma_2})^T$. Here we have defined $v_i^{\Gamma_1}$ as a scalar, however it could also be a vector value $\boldsymbol{v}_i^{\Gamma_1} \in \mathbb{R}^d$.

There are three data mapping options available in preCICE: nearest-neighbor mapping, nearest-projection mapping, and radial basis function mapping. All data mapping variants can be formulated as a linear mapping

$$\boldsymbol{v}^{\Gamma_2} = \boldsymbol{M}\boldsymbol{v}^{\Gamma_1},$$

with a matrix $\boldsymbol{M} \in \mathbb{R}^{N_{\Gamma_2} \times N_{\Gamma_1}}$. Regardless of the mapping method, we define two types of mapping on top of the mapping algorithm: consistent and conservative. Consistent mapping exactly reproduces constant functions on $\boldsymbol{\Gamma}_1$ and $\boldsymbol{\Gamma}_2$. This is typically used for physical data fields such as displacement, velocity, pressure or stress. Conservative mapping preserves the integral of the values on the input mesh $\boldsymbol{\Gamma}_1$. This conserves integral variables such as forces. Consistent mapping requires that the sum of entries in each row of $\boldsymbol{M}$ equals to one, whereas the mapping is conservative if the sum of entries in each column equals to one. Therefore, conservative mapping can be performed by taking the transpose $\boldsymbol{M}^T$ of a mapping matrix $\boldsymbol{M}$ stemming from a consistent mapping.

2.3.1 Projection-Based Data Mapping

Nearest-neighbour: The nearest-neighbour consistent mapping identifies the closest vertex $\boldsymbol{x}_i^{\Gamma_1} \in \boldsymbol{\Gamma}_1$ for each vertex $\boldsymbol{x}_j^{\Gamma_2} \in \boldsymbol{\Gamma}_2$ based on the Euclidean distance between the vertices. The value $v_j^{\Gamma_2} \in \boldsymbol{\Gamma}_2$ is simply copied from the closest vertex. This is the simplest and easiest method to integrate into a partitioned coupling library. However, it is only first order accurate with regard to the input mesh width. In the case of matching meshes, the nearest neighbor method is accurate and extremely fast. For conservative mapping, each vertex on the input mesh $\boldsymbol{x}_i^{\Gamma_1} \in \boldsymbol{\Gamma}_1$ is mapped to the output mesh $\boldsymbol{x}_j^{\Gamma_2} \in \boldsymbol{\Gamma}_2$. If multiple input mesh vertices are mapped to a single output mesh vertex, the output value $v_j^{\Gamma_2} \in \boldsymbol{\Gamma}_2$ is simply the sum of the input mesh values, thus conserving the integral of the input mesh $\boldsymbol{\Gamma}_1$. An illustration is provided in Figure 2.6 for both consistent and conservative mapping with the nearest-neighbour method.

Nearest-projection The nearest-projection mapping requires additional mesh connectivity of the interface mesh and is not suitable for point cloud data only. The closest input mesh surface element $e_i^{\Gamma_1}$ for each vertex $\boldsymbol{x}_j^{\Gamma_2} \in \boldsymbol{\Gamma}_2$ at the output mesh is identified. The vertex $\boldsymbol{x}_j^{\Gamma_2}$ is projected onto the nearest element $e_i^{\Gamma_1}$ to find the projection point $p(\boldsymbol{x}_j^{\Gamma_2})$. The default for this nearest element is a triangle. If no triangle exists for which the orthogonal projection of the target vertex $\boldsymbol{x}_j^{\Gamma}$ is within the triangle, then the projection is performed to the nearest edge. If no such edge exists, then the nearest-projection reverts back to nearest-neighbor mapping for the respective output vertex. After the vertex projection, a barycentric interpolation (for triangles), or linear interpolation (for edges), is performed to determine the value at $\boldsymbol{x}_j^{\Gamma_2} \in \boldsymbol{\Gamma}_2$. This corresponds to a first order accurate constant interpolation orthogonal to the triangle or edge (projection step) and a second order interpolation within the triangle or edge. The accuracy of the mapping is therefore dependent on the ratio of these two error components, projection vs. interpolation error. For

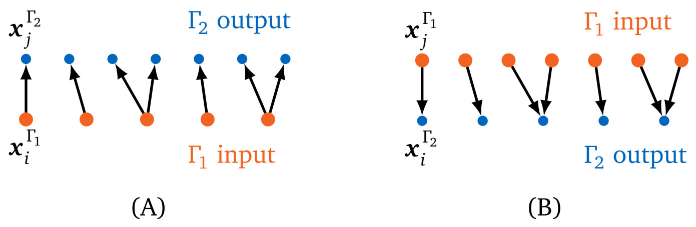

$\boldsymbol{x}_j^{\Gamma_2}$
Γ_2 output
$\boldsymbol{x}_i^{\Gamma_1}$
Γ_1 input
(A)
$\boldsymbol{x}_j^{\Gamma_1}$
Γ_1 input
$\boldsymbol{x}_i^{\Gamma_2}$
Γ_2 output
(B)

FIGURE 2.6 Nearest-neighbor mapping variants for consistent (left) and conservative mapping (right). In consistent mapping, each vertex $\boldsymbol{x}_j^{\Gamma_2}$ in the output mesh attains the value from its closest neighbour $\boldsymbol{x}_{i(j)}^{\Gamma_1}$ on the input mesh. In the conservative variant, each input vertex $\boldsymbol{x}_i^{\Gamma_1}$ adds its value to its nearest neighbour $\boldsymbol{x}_{j(i)}^{\Gamma_2}$ at the output mesh, such that output vertices are assigned either zero, a single value from the input mesh or a sum of values from several vertices of the input mesh.

meshes on a flat surface, there is typically no deviation of the vertex normal to the surface plane, and therefore no projection error. In this case, second order accurate mapping with respect to the edge length can be achieved. However, for more realistic scenarios, in particular for meshes of differing refinement on curved surfaces, the projection error can be the dominant factor. An example of the nearest-projection mapping is shown in Figure 2.7.

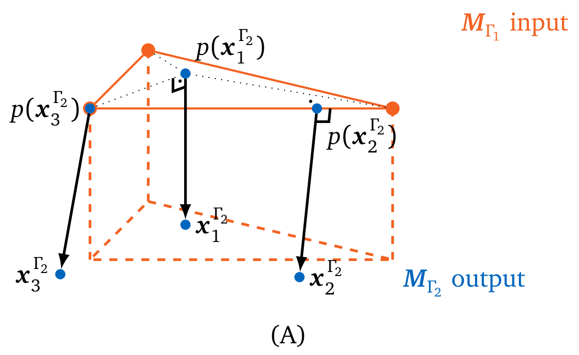
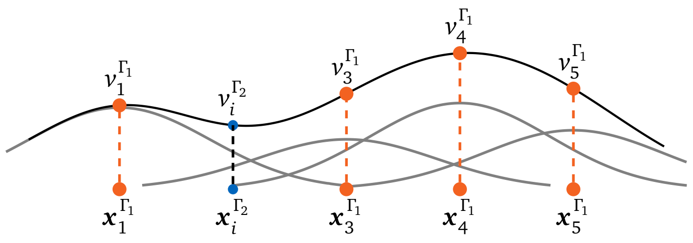

M_{Γ_1} input
$p(\boldsymbol{x}_1^{\Gamma_2})$
$p(\boldsymbol{x}_3^{\Gamma_2})$
$p(\boldsymbol{x}_2^{\Gamma_2})$
$\boldsymbol{x}_1^{\Gamma_2}$
$\boldsymbol{x}_3^{\Gamma_2}$
$\boldsymbol{x}_2^{\Gamma_2}$
M_{Γ_2} output
(A)

FIGURE 2.7 Nearest-projection mapping to the nearest surface, edge and vertex. The orthogonal projection is performed in the opposite direction to the arrow, with the data mapping in the direction of the arrow. The data mapping information transfer is shown for a point onto a triangle $\boldsymbol{x}_1^{\Gamma_2} \leftarrow p(\boldsymbol{x}_1^{\Gamma_2})$, an edge $\boldsymbol{x}_2^{\Gamma_2} \leftarrow p(\boldsymbol{x}_2^{\Gamma_2})$, and a vertex $\boldsymbol{x}_3^{\Gamma_2} \leftarrow p(\boldsymbol{x}_3^{\Gamma_2})$.

2.3.2 Radial Basis Function Interpolation

A popular alternative to the projection-based methods is radial basis function (RBF) interpolation. RBFs come from the field of approximation theory, where complex functions can be represented by a series of simpler ones. In the context of multi-physics simulations, RBFs usually allow for higher interpolation accuracy between non-matching meshes at the coupling interface, while only



$v_1^{\Gamma_1}$
$v_i^{\Gamma_2}$
$v_3^{\Gamma_1}$
$v_4^{\Gamma_1}$
$v_5^{\Gamma_1}$
$\boldsymbol{x}_1^{\Gamma_1}$
$\boldsymbol{x}_i^{\Gamma_2}$
$\boldsymbol{x}_3^{\Gamma_1}$
$\boldsymbol{x}_4^{\Gamma_1}$
$\boldsymbol{x}_5^{\Gamma_1}$

FIGURE 2.8 Radial basis function interpolation using Gaussian basis functions in 1D. The known values at $v_i^{\Gamma_1}$ are used to determine the interpolation function $\boldsymbol{s}(\boldsymbol{x})$ (black line). The interpolation function is formed by the summation of smaller Gaussian basis functions (gray lines) during the compute step. The interpolation step determines the values $v_i^{\Gamma_2}$ at all locations $\boldsymbol{x}_i^{\Gamma_2}$.

requiring point cloud information of the coupling interface.

RBFs use radially symmetric basis functions centered at the vertices $\boldsymbol{\Gamma}_1 = \{\boldsymbol{x}_1^{\Gamma_1}, \ldots, \boldsymbol{x}_{N_{\Gamma_1}}^{\Gamma_1}\}$ for a solver $\boldsymbol{S}_1$. The global RBF interpolant $\boldsymbol{s} : R^d \to R$ is given by the sum of radially symmetric basis functions $\varphi_i(\boldsymbol{x}) := \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i^{\Gamma_1}\|, \xi)$ with a given $\varphi : \mathbb{R} \to \mathbb{R}$.

$$s(\boldsymbol{x}) = \sum_{j=1}^{N_{\Gamma_1}} \gamma_j \cdot \varphi\left(\| \boldsymbol{x} - \boldsymbol{x}_j^{\Gamma_1} \|_2, \xi\right) + q(\boldsymbol{x}), \tag{2.31}$$

where the global first order polynomial $q(\boldsymbol{x}) = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$ is added to ensure that constant and linear functions are interpolated exactly and ξ is the basis function shape parameter. The RBF interpolation method finds the set of coefficients $\boldsymbol{\gamma}_i \in R, i = 1, ..., N_{\Gamma_1}$ that fulfil the interpolation condition

$$s(\boldsymbol{x}_i^{\Gamma_1}) = v_i^{\Gamma_1}, \quad \forall i = 1, ..., N_{\Gamma_1}. \tag{2.32}$$

The under-determined system requires regularisation by the polynomial condition

$$\sum_{i}^{\boldsymbol{\Gamma}_1} \gamma_i \cdot p(\boldsymbol{x}_i^{\boldsymbol{\Gamma}_1}) = 0 \tag{2.33}$$

for the polynomial $p : \mathbb{R}^3 \to \mathbb{R}$ of degree less than or equal to $q(\boldsymbol{x})$. A 1D example of the RBF interpolation is shown in Figure 2.8, ignoring the existence of the polynomial $q(\boldsymbol{x})$ for simplicity of the illustration. The summation of basis functions forms the interpolation function $s(\boldsymbol{x})$, which can be used to determine the interpolation value at any point $\boldsymbol{x}'$. If the point $\boldsymbol{x}'$ falls outside the domain of the input mesh $\boldsymbol{\Gamma}_1$, then an extrapolation is performed, which often results in a lower accuracy than interpolation. Therefore, the point $\boldsymbol{x}'$ should always fall within the domain of the interpolation.

The basis functions and the polynomial can be combined into a linear system of equations stemming from the interpolation conditions in Equation 2.32 and the additional condition in Equation 2.33:

$$\underbrace{\begin{bmatrix} \boldsymbol{M}_{\Gamma_1,\Gamma_1} & \boldsymbol{P}_{\Gamma_1} \\ \boldsymbol{P}_{\Gamma_1}^T & \boldsymbol{0} \end{bmatrix}}_{=:\boldsymbol{A}} \underbrace{\begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{bmatrix}}_{=:\tilde{\boldsymbol{\gamma}}} = \underbrace{\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}}_{=:\tilde{\boldsymbol{f}}}, \tag{2.34}$$

where the components of $\boldsymbol{A}$ are the basis functions:

$$\boldsymbol{M}_{\Gamma_1,\Gamma_1} = \begin{pmatrix} \phi(\|\boldsymbol{x}_0 - \boldsymbol{x}_0\|_2, \xi) & \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_0\|_2, \xi) & \cdots & (\|\boldsymbol{x}_{N_{\Gamma_1}} - \boldsymbol{x}_0\|_2, \xi) \\ \phi(\|\boldsymbol{x}_0 - \boldsymbol{x}_1\|_2, \xi) & \cdots & \cdots & (\|\boldsymbol{x}_{N_{\Gamma_1}} - \boldsymbol{x}_1\|_2, \xi) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\boldsymbol{x}_0 - \boldsymbol{x}_{N_{\Gamma_1}}\|_2, \xi) & \cdots & \cdots & \phi(\|\boldsymbol{x}_{N_{\Gamma_1}} - \boldsymbol{x}_{N_{\Gamma_1}}\|_2, \xi) \end{pmatrix},$$

and the polynomial conditions:

$$\boldsymbol{P}_{\Gamma_1} = \begin{pmatrix} 1 & x_{1,1}^{\Gamma_1} & x_{1,2}^{\Gamma_1} & x_{1,3}^{\Gamma_1} \\ 1 & x_{2,1}^{\Gamma_1} & x_{2,2}^{\Gamma_1} & x_{2,3}^{\Gamma_1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N_{\Gamma_1},1}^{\Gamma_1} & x_{N_{\Gamma_1},2}^{\Gamma_1} & x_{N_{\Gamma_1},3}^{\Gamma_1} \end{pmatrix}.$$

Once the coefficients $\gamma_i, \forall i = 1, ..., N_{\Gamma_1}$ and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_3)^T, \in R^4$ have been determined, the interpolant can be evaluated for all points on mesh $\boldsymbol{\Gamma}_2$ by

$$v_j^{\Gamma_2} = s\left(\boldsymbol{x}_j^{\Gamma_2}\right) = \sum_{i=1}^{N_{\Gamma_1}} \gamma_i \cdot \varphi\left(\| \boldsymbol{x}_j^{\Gamma_2} - \boldsymbol{x}_i^{\Gamma_1} \|_2, \xi\right) + q\left(\boldsymbol{x}_j^{\Gamma_2}\right), \quad \forall j = 1, \ldots, N_{\Gamma_2} \tag{2.35}$$

or in matrix format

$$\begin{bmatrix} \boldsymbol{V}_{\Gamma_2} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{P}_{\Gamma_2} & \boldsymbol{M}_{\Gamma_1,\Gamma_2} \end{bmatrix}}_{=:\boldsymbol{C}} \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{\gamma} \end{bmatrix}. \tag{2.36}$$

The components $\boldsymbol{M}_{\Gamma_1,\Gamma_2} \in R^{N_{\Gamma_2} \times N_{\Gamma_1}}$ and $\boldsymbol{P}_{\Gamma_2} \in R^{N_{\Gamma_2} \times 4}$ are defined as

$$\boldsymbol{M}_{\Gamma_1,\Gamma_2} = \begin{pmatrix} \varphi(\|\boldsymbol{x}_0 - \boldsymbol{x}_0\|_2, \xi) & \cdots & \varphi(\|\boldsymbol{x}_{N_{\Gamma_1}} - \boldsymbol{x}_0\|_2, \xi) \\ \vdots & \ddots & \vdots \\ \varphi(\|\boldsymbol{x}_0 - \boldsymbol{x}_{N_{\Gamma_2}}\|_2, \xi) \cdots & \cdots & \varphi(\|\boldsymbol{x}_{N_{\Gamma_1}} - \boldsymbol{x}_{N_{\Gamma_2}}\|_2, \xi) \end{pmatrix},$$

and

$$P_{\Gamma_2} = \begin{pmatrix} 1 & x^{\Gamma_2}_{1,1} & x^{\Gamma_2}_{1,2} & x^{\Gamma_2}_{1,3} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x^{\Gamma_2}_{N_{\Gamma_2},1} & x^{\Gamma_2}_{N_{\Gamma_2},2} & x^{\Gamma_2}_{N_{\Gamma_2},3} \end{pmatrix}.$$

Basis Functions:

A variety of basis functions exist in order to perform the interpolation. The respective choices have a significant impact on the stability and accuracy of the interpolation. Common basis functions are shown in Table 2.2, with the graphs plotted in Figure 2.9. Basis functions are typically divided into two types: global or local basis functions. For global basis functions, the interpolant value at each vertex of the output mesh is influenced by the values of every vertex in the input mesh. Whereas for local basis functions, the value at each vertex of the output mesh is only influenced by input mesh vertices within a specified radius. Most of the different choices for basis functions in Table 2.2 can be adjusted in their exact shape (or width) with the additional support radius ξ. This input parameter influences the stability and accuracy of the interpolation results and is often the first variable to be adjusted to enhance stability or accuracy.

Various numerical methods are available to improve the stability and accuracy of the RBF interpolation. An in-depth study of these methods is provided by Lindner [Lin19]. However, we briefly explain the methods for solving the RBF system, as well as discuss the RBF matrix conditioning and integrated vs. separated polynomial here as we will refer to them later in this work.

TABLE 2.2 Common basis functions used for radial basis function interpolation. For global basis functions, their support is $\mathbb{R}$. Therefore, the sphere of influence of $\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i^{\Gamma_1}\|, \xi)$ covers the whole mesh. Local basis functions have a support radius instead of a shape parameter, where we set $\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i^{\Gamma_1}\|, \xi) = 0$ for $\|\boldsymbol{x} - \boldsymbol{x}_i^{\Gamma_1}\| > \xi$, resulting in a sparse interpolation system matrix $\boldsymbol{A}$ in Equation 2.34. The local basis functions are defined in the range $\zeta \in [0,1]$, where $\zeta = x/\xi$.

Basis Function	Formulation	Support Field
Gaussian	$\exp^{-(\xi\|x\|)^2}$	Global
Thin Plate Splines	$\|x\|^2 \ell og(\|x\|)$	Global
Multiquadratics	$\sqrt{\xi^2 + \|x\|^2}$	Global
Compact Polynomial C0	$(1-\zeta)^2$	Local
Compact Polynomial C2	$1 - 30\zeta^2 - 10\zeta^3 + 45\zeta^4 - 6\zeta^5 - 60\zeta^3 \ell og \zeta$	Local
Compact Polynomial C6	$(1-\zeta)^8(32\zeta^3 + 25\zeta^2 + 8\zeta + 1)$	Local

2.3.3 Solution of the RBF System

The RBF interpolation step can be simplified as the solution of a linear problem

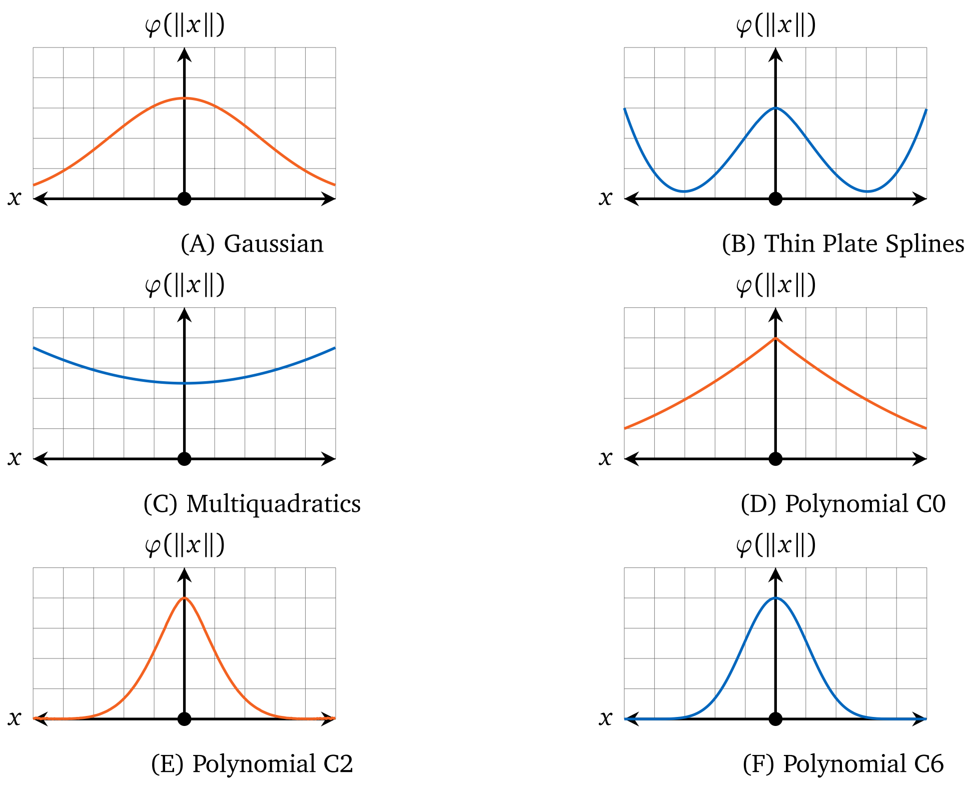


FIGURE 2.9 Various basis functions implemented in preCICE. The Gaussian, thin plate splines (TPS) and multiquadratic basis functions are global, however only TPS is parameter free. The polynomial basis functions all require a cutoff radius after which the basis function equals to zero. Likewise, the Gaussian basis function can be local by applying the same cutoff threshold.

$$A\tilde{\gamma} = \tilde{f}, \tag{2.37}$$

from Equation 2.34. The solution $\tilde{\gamma}$ can be found using either direct or iterative solvers. This is realised in preCICE by using Eigen [Gue10] for direct solvers, and PETSc [Bal22] for iterative solvers. The exact choice of a solver influences the accuracy of the RBF interpolation, runtime and the maximum size of the input mesh allowed.

Eigen has a range of direct solvers that utilise various decompositions, including lower-upper (LU), QR and SVD decompositions. Within preCICE, a QR-decomposition is used as a direct solver, where $\boldsymbol{A} = \boldsymbol{QR}$ and the solution of the RBF interpolation is computed by

$$\tilde{\gamma} = -\boldsymbol{R}^{-1}\boldsymbol{Q}^{T}\tilde{f}. \tag{2.38}$$

As matrix $\boldsymbol{A}$ is square, computing the QR-decomposition is considered to be of order $\mathcal{O}(N_{\Gamma_1}^3)$ in time and $\mathcal{O}(N_{\Gamma_1}^2)$ in memory. For iterative solvers, PETSc offers a range of solvers that are suited for different degrees of sparsity. By default, preCICE uses the GMRES solver. An iterative solver finds the solution to Equation 2.37 by successively iterating the values of $\boldsymbol{\gamma}$ until

$$\|\boldsymbol{A}\boldsymbol{\gamma}_j - \boldsymbol{f}\|_2 < \epsilon_{iterative}, \tag{2.39}$$

where $\boldsymbol{\gamma}_j$ is solution at iteration j and $\epsilon_{iterative}$ is a user defined iterative solver threshold. The GMRES solver has a computational complexity of $\mathcal{O}(N_{\Gamma_1} j^2)$ in time and $\mathcal{O}(N_{\Gamma_1} j)$ in memory.

The two classes of methods, direct and iterative, affect the simulation runtime in different ways. The direct solver incurs a single very expensive *mapping computation* step, where the QR-decomposition is performed. However, each *RBF evaluation* is extremely cheap, requiring only a single solve of Equation 2.38. Therefore, the computational expense is large during the simulation initialisation, and cheap in each coupling iteration.

The opposite is true for the iterative solver. The initialisation phase requires building the PETSc iterative matrices, which requires several parallel communication steps, which is less costly than a QR-decomposition. However, each coupling iteration requires calculating the coefficients $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ of the RBF mapping in potentially hundreds of iterations of the GMRES solver in Equation 2.39. In short, and perhaps naive description, the direct solver has an expensive computation but cheap evaluation, while the iterative solver has a cheap computation and expensive evaluation. The *mapping computation* as implemented in preCICE for PETSc is described in [Uek16]; [Lin19].

2.3.4 Conditioning and Separated Polynomial

A well conditioned interpolation matrix $\boldsymbol{A}$ is required in order to achieve accurate interpolation. The condition number κ of the interpolation matrix is a measure for the sensitivity of the solution to changes in the input for a system. It is also an important metric for the difficulty to solve the system of equations (numerical stability of direct solvers, and number of iterations of iterative solvers).

If using the compact polynomial that has a local support radius, increasing the support radius results in a decrease in the theoretical interpolation error while increasing condition number. However, a trade-off is eventually observed between increasing condition number and decreasing interpolation error. Consequently, there is a turning point, after which the increasing theoretical accuracy is deteriorated by the numerical instability in solving the system. Due to this trade-off, RBF interpolation requires careful adaptation of the parameters of the respective basis. An analysis of the condition number versus interpolation accuracy was performed by Lindner [Lin19].

An effective method to reduce the condition number, and improve the interpolation accuracy, is to use separated polynomial interpolation. The RBF interpolation matrix $\boldsymbol{A}$ in Equation 2.34 comprises the basis function evaluations in $\boldsymbol{M}$ and polynomial terms $\boldsymbol{P}$. Global basis functions result in a dense matrix $\boldsymbol{A}$, whereas local basis functions result in a sparse matrix $\boldsymbol{A}$, where the smaller the support radius, the sparser $\boldsymbol{A}$ becomes. However, $\boldsymbol{P}$ is always dense and, thus, limits the sparsity of $\boldsymbol{A}$, influencing the method of solving the system of equations.

Lindner [Lin19] implemented the separated polynomial, where the polynomial is solved separately by solving a least-squares regression problem

$$\min \|\boldsymbol{P}_{\Gamma_1}\boldsymbol{\beta} - \boldsymbol{f}\|$$

to determine the polynomial coefficients $\boldsymbol{\beta}$. Subsequently, a modified interpolation problem

$$\boldsymbol{M}_{\Gamma_1,\Gamma_1}\boldsymbol{\gamma} = \boldsymbol{f} - \boldsymbol{P}_{\Gamma_2}\boldsymbol{\beta}$$

is solved to determine the coefficients $\boldsymbol{\gamma}$ of the RBF basis. The output is computed in the same manner as Equation 2.36,

$$\boldsymbol{s} = \boldsymbol{M}_{\Gamma_1,\Gamma_2}\boldsymbol{\gamma} + \boldsymbol{P}_{\Gamma_2}\boldsymbol{\beta}.$$

Splitting $\boldsymbol{M}$ and $\boldsymbol{P}$ allows for the use of individual solvers specific to the requirements of the system to be solved.

2.3.5 Limitations of Radial Basis Functions

Barriers to widespread usage of RBF interpolation are stability issues and significant computational cost. For direct solvers, RBFs may grow with $\mathcal{O}(N_{\Gamma_1}^3)$ in terms of computational cost and $\mathcal{O}(N_{\Gamma_1}^2)$ in memory. This typically restricts the input mesh to have less than 10^4 vertices for most practical cases. Therefore, indirect solvers are often used for a large number of vertices, but this introduces new limitations.

Multiple parameters are critical to the performance of the RBF interpolation. Firstly, the `solver-rtol` defines the relative tolerance $\epsilon_{iterative}$ in Equation 2.39. By selecting a smaller tolerance, a higher accuracy can be achieved, however the solver will struggle to solve the linear system, requiring longer evaluation times, or in the worst case, failing to achieve the required solver tolerance at all. The second influential parameter is the width of the basis functions (compact `support-radius` for local basis functions), which determines the sphere of influence around each vertex (determined by ξ in Table 2.2). An exception is the Gaussian basis function, which requires additional steps to create a local basis function. The Gaussian shape parameter ξ is determined such that a value close to zero (default is 10^{-9} in preCICE) occurs at the `support-radius` edge, after which $\varphi(\|\boldsymbol{x}\|) = 0$ for any point further than the `support-radius`. Typically, the larger the sphere of influence, the better the interpolation accuracy. However, this also comes with larger memory requirements, longer compute times and worsening of the matrix conditioning [Lin19]. In order to improve interpolation accuracy, the support-radius, or shape parameter, is the most common variable to optimise.

Another step to improve accuracy and conditioning is to remove the polynomial term $q(\boldsymbol{x})$ from matrix $\boldsymbol{A}$, and solve this separately [Lin19], by setting `polynomial="separate"` in the preCICE configuration file. The last significant parameter is the choice to use the direct solver

FIGURE 2.10 Surface values for the Elastic-Tube-3D test case at the middle (t = 0.005s) and the end (t = 0.01s) of the simulation runtime. At the beginning of the simulation, the front-left part of the surface has the larger magnitude values on the coupling interface, whereas this shifts towards the back-right part at the end of simulation. Determining when the shape parameter optimisation needs to be performed is non-trivial for time-dependent problems and cannot be known before-hand.

with `use-qr-decomposition="1"` in the preCICE configuration file. This allows for a single, expensive computation step, followed by fast evaluation of the RBF interpolation step in each coupling iteration during the simulation runtime, and removing the dependence on the parameter `solver-rtol`. This makes a direct solver attractive for RBF mapping in multi-physics simulations, as potentially thousands of RBF evaluations may be required. Therefore, it may be feasible to compute a single, expensive, QR-decomposition. The attractiveness of the direct solver is over-shadowed by the extreme computational cost of the interpolation computation step, limiting the allowed mesh size.

Multi-physics simulations also have a unique limitation on optimising the shape parameter. Previous studies always optimise the shape parameter based on either knowing the test function or using the known values at the input mesh. Any method of optimising the shape parameter is costly and would introduce the costly shape parameter optimisation procedure many times throughout the coupled simulation. Surface values for the perpendicular-flap test case at two different time steps are shown in Figure 2.10, showing how the surface values can change even for such simple examples. Whether or not shape parameter optimisation would be necessary for this case is unknown, and therefore brings into doubt the practically of mid-simulation shape parameter optimisation.

A typical RBF parameter setup is shown in Figure 2.11, where the `solver-rtol`, `polynomial`, `support-radius` and `basis function` variables are shown.

In Chapter 4, an evaluation of the sources of errors and instabilities is presented. This is followed by methods to reduce the computational cost with creating the interpolation matrix $\boldsymbol{A}$ and to simplify the selection of the shape parameter. Numerical testing is performed to establish the computational performance advantage of the new RBF interpolation method, and to find a set of good default shape parameter/support radius values for the RBF basis functions.

```
<mapping:rbf-compact-polynomial-c6
          shape-parameter="{float}"
          solver-rtol="1e-09"
          constraint="{string}"
          direction="{string}"
          from="{string}"
          polynomial="separate"
          to="{string}"
          use-qr-decomposition="0"
          x-dead="0" y-dead="0" z-dead="0"/>
```

FIGURE 2.11 Input parameter selection for radial basis function data mapping from the XML reference. A variety of user-specified parameters are required for efficient and robust data interpolation. The necessary parameters and data types are provided if no default value is given. The XML reference is available at `https://precice.org/configuration-xml-reference.html`.

3 Quasi-Newton Methods for Coupling Acceleration

Previous work in multi-physics coupling has shown that quasi-Newton methods are able to provide sufficient acceleration of the fixed-point problem for strongly coupled simulations [Bog16b]; [Uek16]; [Sch18]. However, when used in practice, additional steps are required to improve the robustness and efficiency of the quasi-Newton coupling algorithm. Additional pre-scaling (Section 2.2.3) improves Jacobi type coupling where the coupling residuals may live on different scales, and filtering (Section 2.2.4) improves the condition of $\boldsymbol{V}_k$ by limiting the build-up of linearly dependent columns. However, these measures not only add significant computational overhead to the quasi-Newton method, but they also add additional configuration parameters to the coupling input file. This complexity is passed on to the user, where "bad" input parameters may reduce the performance of the coupled problem.

In the beginning of this chapter, we present enhanced methods to perform pre-scaling (Section 3.1.1), filtering (Section 3.1.2) and IQN-IMVJ reuse enhancements (Section 3.1.3) for multi-physics simulations with preCICE. Section 3.2 describes the typical types of problems found in fluid-structure interaction simulations, a common type of multi-physics simulation. The governing equations of fluid and solid physics solvers[1], numerical experimental setup, software and hardware setups used for the numerical simulations are defined. Section 3.3 presents the results from the numerical testing of the enhanced methods, leading to defining a set of good default values to simplify the input parameter selection in Section 3.4.

[1]In this section, we only perform FSI simulations using and fluid solver (CFD) and a solid solver (CSM).

3.1 Enhancing Partitioned Quasi-Newton Coupling Acceleration

The various input parameters that are required by preCICE, interact with each other in a complex manner. For example, the QR1 filter does not explicitly need to perform a QR-decomposition in each coupling iteration, but it will if pre-scaling is selected. If the QR2 filter is selected, then pre-scaling does not result in any additional computational overhead as a complete QR-decomposition is performed by the QR2 filter. Previous work by Uekermann [Uek16] also found that the QR1 and the QR2 filter require very different filter limit values ϵ_{filter} to filter $\boldsymbol{V}_k$ effectively. However, as discussed in Section 2.2.4, this is likely due to the relative versus absolute criteria for deleting a column between the two filters. Another consideration is how the input parameters selected influence the simulation runtime. For example, selecting large values for the number of time steps ζ and iterations reused η, or selecting a low threshold for ϵ_{svd} in the re-start step of *IQN-IMVJ* may improve the rate of convergence (decreasing simulation runtime), but also add extensive computational overhead (increasing simulation runtime). In this section, we look at the individual computational steps performed during the quasi-Newton acceleration, identify areas for improvement, and propose new methods to increase performance.

The computational complexity of various algorithms required for the quasi-Newton update, QR-decomposition and IQN-IMVJ restart mechanisms are described in detail in the work of Uekermann [Uek16] and Scheufele [Sch18]. Here, we repeat the main findings of the parallel computational complexity of the individual algorithmic components in order to evaluate where computational savings can be found. Examining Table 3.1, we see that the complexity of the QR-update is in the order of $\mathcal{O}\left(\frac{N}{p}\eta^2\right)$ (we assume that $\eta \ll N$) for a single column insertion step, with N degrees of freedom (DoF) decomposed over p ranks (assuming ideal load balancing). This step is required η times per iteration in the case where pre-scaling or the QR2 filter is selected, resulting in $\mathcal{O}\left(\frac{N}{p}\eta^3\right)$ for a full QR-decomposition. If the QR-decomposition is reconstructed in each iteration, a significant computational cost is expected when using the IQN-ILS method for a large number of iterations and time steps reused. The computational complexity for the `Backward Substitution`, `Pseudo-inverse`, `Update` $\tilde{w}$, and the `Newton Update` building blocks cannot easily be reduced, and we therefore do not examine these steps further. However, these scale relatively well compared to the QR-decomposition and restart algorithms. Both the RS-SVD and RS-LS algorithms scale with $\mathcal{O}\left(\frac{N}{p}\kappa^2\right)$ and $\mathcal{O}\left(\frac{N}{p}\eta_m^2\right)$, where κ is the number of dominant modes retained in the truncated SVD, and η_m is the total number of columns from the last m time steps in Equation 2.26, i.e., the sum of all columns in all $\boldsymbol{V}_k$-matrices in the sum representation of $\boldsymbol{J}_{prev}^{-1}$ before restart. It is important to remember that the restart algorithm is only performed once every m time steps, whereas the expensive QR-decomposition is performed in each iteration. Therefore, there can be a great potential to reduce the computational cost of the QR-decomposition in the *IQN-ILS* by a new combination of pre-scaling and filtering that allows to maintain both the good convergence rate achieved with frequent pre-scaling, and the cheap QR-update of the quasi-Newton without pre-scaling. For the *IQN-IMVJ*, a smart restart strategy with pre-scaling only when required can reduce the number of iterations for parallel coupling over the current approach, where pre-scaling weights are frozen after the first time step.

In the following section, we evaluate the pre-scaling and filtering methods of Section 2.2.3 and Section 2.2.4, respectively, and present new algorithms to reduce the computational overhead while maintaining comparable performance in terms of the convergence rate and robustness. Furthermore, reducing the restriction of the size of $\boldsymbol{V}_k$ (due to the QR-decomposition cost) allows the *IQN-IMVJ* to reuse multiple previous time steps in each matrix group in $\sum_{q=0}^{n} \widetilde{\boldsymbol{W}}_{k_q}(\boldsymbol{V}_{k_q}^{\dagger}\boldsymbol{r}^k)$.

The developed enhancements are divided into three parts:

1. Firstly, a new pre-scaling monitoring method is developed. The pre-scaling weights are frozen at the end of the first time step. In each following iteration, the pre-scaling weights are evaluated in each coupling iteration and are only updated if required according to a suitable criterion. This eliminates the need to perform a complete QR-decomposition in each coupling iteration, but also allows the scaling weights to adapt to changing behaviour of the solver's residuals.
2. Secondly, a new QR3 filter is introduced that mimics the behaviour of the QR2 filter without performing a complete QR-decomposition in each iteration.
3. Finally, a modification of the IQN-IMVJ method is presented, where $\boldsymbol{J}_{prev}^{-1}$ is not updated in every time step, such that the columns of more than one time step are in the current $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$, allowing the QR3 filter to potentially remove outdated information.

These methods are not intended to be used in isolation, but rather supplement each other to improve the runtime of the acceleration scheme. The new pre-scaling weight monitoring presented below can be used independently, but the new QR3 filter requires that either the pre-scaling weights are frozen entirely, or that the pre-scaling monitoring is activated. The modification of the *IQN-IMVJ* method can be used with the QR2 filter, but significant computational gains will only be made with the combined pre-scaling monitoring and QR3 filter.

3.1.1 Pre-scaling Weight Monitoring

As described in Section 2.2.3, the pre-scaling weights $\boldsymbol{\Lambda}_k$ are updated in each coupling iteration, which makes a complete QR-decomposition of $\boldsymbol{V}_k$ necessary in each iteration. Here, we introduce a pre-scaling weight monitoring method that tracks the value of the pre-scaling weight values and decides when to re-scale based on the monitoring conditions. If the pre-scaling weights are not updated, a QR-update column insertion step is sufficient to update the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, otherwise a complete QR-decomposition must be performed.

For all iterations in the first time step, the pre-scaling weights $\bar{\boldsymbol{\Lambda}}_k = (\bar{\lambda}_{k,1}, \bar{\lambda}_{k,2})^T$ are updated in each coupling iteration, after which they are frozen at the end of the time step. As the first time step involves the initial establishing of quasi-Newton information, in particular $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ may start with physically invalid initial data, a lot of changes are in general expected in each iteration. Therefore, updating the weights in each iteration during the first time step may help improve stability and convergence. From the start of the second time step, the theoretical values of the

TABLE 3.1 Computational complexity of various algorithms during the quasi-Newton acceleration update. The computational complexity is given in terms of the number of interface degrees of freedom N, computational ranks p, number of columns η in $\boldsymbol{V}_k$, number of time steps for each restart m, and the number of columns for the RS-LS restart η_m, and number of SVD modes retained κ. Similar tables can be found in Uekermann [Uek16] and Scheufele [Sch18].

Algorithm	IQN-ILS	IQN-IMVJ
QR-Update	$\mathcal{O}(\eta^2 \frac{N}{p}) + \mathcal{O}(\eta^3)$	$\mathcal{O}(\eta^2 \frac{N}{p}) + \mathcal{O}(\eta^3)$
Back-Sub.	$\boldsymbol{R}\alpha = -\boldsymbol{Q}^T \boldsymbol{r}^k$: $\mathcal{O}(\eta \frac{N}{p}) + \mathcal{O}(\eta^2)$	–
Pseudo-inverse	–	$\boldsymbol{R}\boldsymbol{V}^\dagger = \boldsymbol{Q}^T$: $\mathcal{O}(\eta^2 \frac{N}{p})$
Newton Update	$\Delta \boldsymbol{x} = \boldsymbol{W}_k \alpha$: $\mathcal{O}(\eta \frac{N}{p})$	$\Delta \boldsymbol{x} = \sum_{q=n-m}^{n} \widetilde{\boldsymbol{W}}_{kq} (\boldsymbol{V}_{kq}^\dagger \boldsymbol{r}^k)$: $\mathcal{O}(m\eta \frac{N}{p}) + \mathcal{O}(m\eta \ell o g p)$
Update $\tilde{\boldsymbol{w}}$	–	$\mathcal{O}(\eta \frac{N}{p}) + \mathcal{O}(\eta \ell o g p)$
RS-SVD Update	–	$\mathcal{O}(\kappa^2 (\frac{N}{p} + \eta)) + \mathcal{O}(\frac{\kappa^3}{\eta})$
RS-LS Update	–	$\mathcal{O}(\eta_m^2 \cdot \frac{N}{mp}) + \mathcal{O}(\frac{\eta_m^3}{m})$

new scaling weights $\bar{\boldsymbol{\Lambda}}_k^{new}$ are computed according to Equation 2.28 in each coupling iteration. However, the values of the actual pre-scaling weights $\bar{\boldsymbol{\Lambda}}_k$ are not updated and, therefore, the QR-decomposition does not need to be recomputed. If at least one pre-scaling weight within $\bar{\boldsymbol{\Lambda}}_k^{new}$ differs by more than a specified factor of $\epsilon_{prescale}$ from the corresponding weight in $\bar{\boldsymbol{\Lambda}}_k$, then the pre-scaling weights are updated. The requirement whether the pre-scaling weights should be updated is, thus, defined as

$$\exists i \quad \text{with} \quad \bar{\lambda}_{k,i}^{new} > \epsilon_{prescale} \cdot \bar{\lambda}_{k,i} \qquad or \qquad \bar{\lambda}_{k,i}^{new} < \frac{\bar{\lambda}_{k,i}}{\epsilon_{prescale}}. \tag{3.1}$$

If the condition of equation 3.1 is true, then the pre-scaling weights are updated $\bar{\boldsymbol{\Lambda}}_k := \bar{\boldsymbol{\Lambda}}_k^{new}$. Only if the update is performed, a complete re-computation of $\boldsymbol{Q}$ and $\boldsymbol{R}$ is triggered. The purpose of the pre-scaling is not just to normalise each sub-vector of residuals $\boldsymbol{r}_i^k$, but to instead bring all sub-vectors into the same order of magnitude. Therefore, the ratio between each sub-vector is more important than the magnitude themselves. However, by examining each sub-vector independently in equation 3.1, the pre-scaling weight monitoring method generalises to an arbitrary number of solvers with an arbitrary number of sub-vectors.

By only updating the pre-scaling weights periodically, fewer complete QR-decomposition are required. A technically simpler method would be to freeze the weights after a specified time, or to periodically update the weights. However, freezing the weights does not allow for the sub-vectors to be scaled appropriately under changing conditions for time-dependent problems, and periodically updating the weights adds the length of the pre-scaling update period as another parameter for tuning.

3.1.2 New QR3 Filter

Section 2.2.4 described the QR filtering procedures for the QR1 and QR2 filters. There are small, but significant differences between the two. QR1 uses the absolute vector norm of an orthogonalised column as a deletion criterion, whereas QR2 uses a criterion relative to the norm of the original column. QR1 allows for deleting columns and proceeding to the next coupling iteration using only cheap QR-update steps, whereas QR2 requires a complete re-calculation of the QR-decomposition in each quasi-Newton iteration. Combined with the pre-scaling weight monitoring, computational savings can be made with the QR1 filter, as fewer QR-decompositions would be performed, and only column insertions/deletions would suffice. However, the QR1 filter is highly dependent on the scaling of the problem at hand, whereas the QR2 filter, with its relative criterion, is scaling independent.

For classical, medium-scaled surface coupled simulations, the QR-decomposition was considered to have an almost negligible computational cost due to the assumption that the number of DoF on the coupling interface, N, is much smaller than the number of DoF within the solvers, N_{solver}. However, this cost is not always negligible due to the following reasons:

1. the number of columns η retained in $\boldsymbol{V}_k$ may grow very large for simulations with many unstable modes in the coupling equation, and the cost of inserting a column into QR has a computational complexity of $\mathcal{O}(N\eta^2)$,
2. for volume coupled problems, the number of interface DoF is equal to the number of all DoF in the domain, i.e., $N = N_{solver}$, and is not negligible in size.

Under these conditions, reducing the number of QR-decompositions for the QR2 filter is imperative.

To understand where to improve the QR2 filter, we consider the possible outcomes of the filter step in each iteration. If the QR2 filter does not remove any column from $\boldsymbol{Q}$ and $\boldsymbol{R}$ during a filter step, $\boldsymbol{Q}$ and $\boldsymbol{R}$ will be the same before and after the filter step. Therefore, the complete QR-decomposition is unnecessarily expensive in this scenario as it did not perform any filtering. In this work, we introduce the QR3 filter, that can provide comparable functionality to the QR2 filter, while potentially being orders of magnitude less computationally expensive. The only requirement for the QR3 filter is that the pre-scaling weights remain constant between two coupling iterations, otherwise a normal QR2 filter step is performed instead to recompute $\boldsymbol{Q}$ and $\boldsymbol{R}$ with the new pre-scaling weights. The QR3 filter is split into three steps:

1. the newest column of $\boldsymbol{V}_k$ is inserted into an existing QR decomposition (see Section 2.2.4),
2. the condition of $\boldsymbol{R}$ is checked to tag any column that would be removed according to the same criterion as the QR2 filter. A column i is tagged for deletion if

$$\boldsymbol{R}_{ii} < \epsilon_f \|\boldsymbol{V}_{k,(:,i)}\|_2.$$

```
Add newest column $\overline{v} = (V_k)_{:,1}$ to $QR$
$filter$ ← false
for $i = \eta, \ldots, 2$ do                    ⇝ Starts from oldest column and works forwards
  if $R_{ii} < \epsilon_f \|(V_k)_{:,i}\|_2$ then
    $filter$ ← true
    break
if $filter$ then
  Compute QR2 Filter Step
```

ALGORITHM 3.1 Pseudo-code for the QR3 Filter. The new column added to $\boldsymbol{V}_k$ is used to update $\boldsymbol{Q}$ and $\boldsymbol{R}$ through a column insertion process. The QR3 filter mimics the filtering check of the QR2 filter without performing a complete QR-decomposition, unless at least one column is tagged for deletion. In this case, a regular QR2 filter step is performed.

3. if at least one column is tagged to be removed, a normal QR2 filter step is performed instead.

The first column $\boldsymbol{R}_{(:,1)}$[2] is not considered for removal as we always want to keep the latest information. This criterion aims to mimic the behaviour of the QR2 filter by using a relative criterion, but without reconstructing $\boldsymbol{QR}$, i.e., without considering the (presumably small) effect of on-the-fly column deletions during the QR-decomposition on the subsequently computed entries of $\boldsymbol{R}$. The QR2 filter only evaluates if a column should be removed once during the construction of $\boldsymbol{Q}$ and $\boldsymbol{R}$, and never re-evaluates the previous columns condition. Therefore, to mimic the QR2 filter without the computational cost, all tagged columns cannot simply be deleted as in the QR1 filter. The *QR3* pseudo-code is shown in Algorithm 3.1.

3.1.3 Information Reuse for Inverse Multi-Vector Jacobian

A significant benefit of the IQN-IMVJ method is the reduced size of $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$, as they only store iterations after the approximation $\boldsymbol{J}_{prev}^{-1}$ has been generated. By default, this occurs every time step in preCICE. This eliminates the number of re-use time steps in $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ as a user defined parameter that requires problem specific tuning. Scheufele [Sch15] studied multi-vector update methods and found that varying the number of time steps stored in $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ does not offer any improvement over storing iterations from one time step only. Furthermore, Scheufele [Sch18] states that the multi-vector update method as presented in Chapter 2, i.e., with one rank-η update in each time step tends to outperform Broyden-like versions with one rank-1 update after each quasi-Newton iteration. This was shown at least for multi-physics applications with slow moving dynamics. However, increasing both the interval between updates of $\boldsymbol{J}^{-1}$ and the rank of the updates $\widetilde{\boldsymbol{W}}_{k_q} \boldsymbol{V}_{k_q}^{\dagger}$ in Equation 2.23, has not been tested before.

As the inverse Jacobian approximation is now a direct summation of previous matrices $\widetilde{\boldsymbol{W}}_{k_q}$ and $\boldsymbol{V}_{k_q}^{\dagger}$, filtering of columns in $\boldsymbol{V}_k$ might be necessary to improve the convergence rate. Previous

[2]subscript refers to the row and column number. $(:,1)$ refers to the leftmost column of R (the newest column of information).

```
for $k = 1, 2, \ldots$ do
  $\tilde{\boldsymbol{x}}^k = \boldsymbol{H}(\boldsymbol{x}^k)$ and $\boldsymbol{r}^k = \tilde{\boldsymbol{x}}^k - \boldsymbol{x}^k$
  if converged then
    break
  Update $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$
  Update $\tilde{\boldsymbol{w}} = \Delta\tilde{\boldsymbol{x}}^k - \sum_{q=0}^{n-1} \widetilde{\boldsymbol{W}}_{k_q}(\boldsymbol{V}_{k_q}^{\dagger}\boldsymbol{r}^k)$      $\rightsquigarrow$ $n-1$ summands in $\boldsymbol{J}_{prev}^{-1}$
  Append $\tilde{\boldsymbol{w}}$ to front of $\widetilde{\boldsymbol{W}}_{k_n}$
  Solve $\boldsymbol{R}\boldsymbol{V}^{\dagger} = \boldsymbol{Q}^T$
  $\Delta\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \sum_{q=0}^{n} \widetilde{\boldsymbol{W}}_{k_q}(\boldsymbol{V}_{k_q}^{\dagger}\boldsymbol{r}^k)$      $\rightsquigarrow$ $n$ summands in $\Delta\boldsymbol{x}^{k+1}$ update
  $\boldsymbol{x}^{k+1} = \tilde{\boldsymbol{x}}^k + \Delta\boldsymbol{x}^{k+1}$
  if Time step converged and $\zeta$ time steps since previous $\boldsymbol{J}_{prev}^{-1}$ update then
    Create a new group of $\boldsymbol{V}_{k_{n+1}}$, $\boldsymbol{W}_{k_{n+1}}$ and $\widetilde{\boldsymbol{W}}_{k_{n+1}}$
    $n := n + 1$
```

ALGORITHM 3.2 Pseudo-code for the modified IQN-IMVJ quasi-Newton methods with time step information reuse. The algorithm assumes that n groups of $\widetilde{\boldsymbol{W}}_{k_n}$ and $\boldsymbol{V}_{k_n}^{\dagger}$ have been created, with $n-1$ groups for the approximation $\boldsymbol{J}_{prev}^{-1}$.

experience has also shown that deleting only a few columns near the beginning of the simulation can vastly improve the robustness and convergence rate of the simulation. However, filtering typically requires that almost linearly dependent columns build-up in $\boldsymbol{V}_k$, which may not occur for small matrices. Increasing the size of $\boldsymbol{V}_k$, i.e., storing iterations from multiple time steps, may increase the likelihood of filtering columns and might offer some improvement to the convergence rate.

Examining table 3.1, the `Update` $\tilde{\boldsymbol{w}}$ and `Newton Update` steps are not limited by the number of iterations η, and once again, the `QR–update` column addition is the limiting factor. However, applying both pre-scaling weight monitoring and the QR3 filter introduced above may offer a solution to this bottleneck.

Therefore, we modified the existing *IQN-IMVJ* method to allow for each group $\boldsymbol{V}_{k_q}$ and $\boldsymbol{W}_{k_q}$ to contain iterations from ζ time steps, which is a user defined parameter. In each iteration during the j^{th} time step[3], a new column is added to the most recent group $\boldsymbol{V}_{k_n}$ and $\boldsymbol{W}_{k_n}$. The new column $\tilde{\boldsymbol{w}}$ is added to $\widetilde{\boldsymbol{W}}_{k_n}$ by

$$\tilde{\boldsymbol{w}} = \Delta\tilde{\boldsymbol{x}}^k - \sum_{q=0}^{n-1} \widetilde{\boldsymbol{W}}_{k_q}(\boldsymbol{V}_{k_q}^{\dagger}\boldsymbol{r}^k)$$

where $n-1$ is the total number of groups (summands) of $\boldsymbol{J}_{prev}^{-1}$. Once columns from ζ time steps have been added to the newest group, the newest groups $\boldsymbol{V}_{k_n}^{\dagger}$ and $\widetilde{\boldsymbol{W}}_{k_n}$ are added to the approximation of $\boldsymbol{J}_{prev}^{-1}$, such that the number of groups grows by 1, and new empty groups $\boldsymbol{V}_{k_{n+1}}$, $\boldsymbol{W}_{k_{n+1}}$ and $\widetilde{\boldsymbol{W}}_{k_{n+1}}$ are created. In similar fashion to the restart methods in Section 2.2.2, the *IQN-*

[3]Note that $n < j$ in the case that ζ time step are stored in each group $\boldsymbol{V}_{k_n}$ and $\boldsymbol{W}_{k_n}$, contrary to that in Equation 2.22. Example, if $\zeta = 2$, then 10 groups will exist after 20 time steps, with each group containing columns from 2 time steps each.

IMVJ method can be restarted after a specified number of groups have been created or a specified number of time steps have occurred. The IMVJ information reuse is similar to the IQN-ILS method, where multiple time steps worth of information is stored in $\boldsymbol{V}_{k_n}$ and $\boldsymbol{W}_{k_n}$, but in this case $\boldsymbol{J}_{prev}^{-1} \neq 0$. The pseudo-code of the information reuse for the IQN-IMVJ method is provided in Algorithm 3.2.

3.2 Numerical Experiments – Setup

In the following section, we define the experimental simulation setup used to test the quasi-Newton enhancements for fluid-structure interaction simulations. Firstly, the governing equations for computational fluid dynamics and computational solid mechanics simulations are briefly described. Next, a variety of test cases are presented, that were selected to test various aspects of the quasi-Newton coupling acceleration. For each simulation, a significant amount of results have been generated. We limit the information presented to show that the new enhancements offer improvements to the quasi-Newton coupling acceleration. Finally, a set of "*good*" default values are suggested, that account for different numerical behaviour of various test cases, while being computationally efficient.

3.2.1 Governing Equations

Multi-physics simulations involve the interplay of various physical phenomena. In this chapter, we specifically focus on fluid-structure interaction simulations. Below, the governing equations relating to the fluid mechanics solvers, solid mechanics solvers, boundary conditions and coupling conditions are described.

Fluid Mechanics: The fluid domain is governed by the Navier-Stokes equations, presented in the arbitrary Lagrangian-Eulerian form defined in the space-time domain $\Omega_F \times [0,T] \in \boldsymbol{R}^d \times \boldsymbol{R}$

$$\rho\left(\frac{\delta \boldsymbol{v}}{\delta t} + ((\boldsymbol{v} - \boldsymbol{v}_m)\cdot\nabla)\boldsymbol{v}\right) = -\nabla p + \mu\Delta\boldsymbol{v} + \rho\boldsymbol{f},$$

$$\nabla\cdot\boldsymbol{v} = 0,$$

where $\boldsymbol{v}$ is the fluid velocity field, p is the pressure field, μ is the shear viscosity, $\boldsymbol{f}$ is the external force acting on the fluid, for example due to gravity, and ρ is the fluid density. As we consider incompressible flow only, ρ is constant. The derivation of the Navier-Stokes formulation and finite volume discretization techniques can be found in [Ver07]. To create a well-posed initial-value problem, suitable boundary conditions are required. The domain boundaries are composed of Neumann boundaries Γ_N and Dirichlet boundaries Γ_D, where $\Gamma = \delta\Omega = \Gamma_D \cup \Gamma_N$. The conditions imposed are

$$\boldsymbol{\sigma}\cdot\boldsymbol{n} = \boldsymbol{f}_N \qquad on \quad \Gamma_N,$$

$$\boldsymbol{v} = \boldsymbol{v}_D \qquad on \quad \Gamma_D,$$

$$\boldsymbol{v}_0(\boldsymbol{x}) = \boldsymbol{v}(t_0, \boldsymbol{x}) \qquad on \quad \Omega.$$

The Neumann boundary condition represents a fixed stress $\boldsymbol{f}_N$ and the Dirichlet boundary represents a fixed velocity of $\boldsymbol{v}_D$. The Cauchy stress tensor $\boldsymbol{\sigma} = -p\boldsymbol{I} + \boldsymbol{\tau}$ is composed of a unidirectional pressure p, a viscous stress tensor $\boldsymbol{\tau}$. We restrict the description to laminar flow only as turbulent flow simulations were not performed in this work.

Solid Mechanics: The formulation found in Wriggers [Wri08] is used to describe the solid mechanics domain. For examples typically found in FSI applications, the solid structure might undergo large deformations and displacements. Hence, a non-linear geometric description of the material is required. More specifically, we use the constitutive equations for large deformation, homogeneous, isotropic, elastic structures. Similar to the fluid solver above, the solid domain is defined in the space time domain $\Omega_S \times [0, T] \in \boldsymbol{R}^d \times \boldsymbol{R}$. Based on an equilibrium of forces, the equation of motion of a solid structure can be defined as

$$\rho\left(\frac{\delta^2 \boldsymbol{u}}{\delta t^2}\right) = \nabla \boldsymbol{S} + \rho \boldsymbol{f},$$

where the second derivative of displacements $\frac{\delta^2 \boldsymbol{u}}{\delta t^2}$ defines the acceleration of a point in the material, $\boldsymbol{f}$ denotes the body forces. The 2^{nd} Piola-Kirchhoff stress tensor $\boldsymbol{S}$ is defined as

$$\boldsymbol{S} = \lambda \cdot tr(\boldsymbol{E})\boldsymbol{I} + 2\mu\boldsymbol{E}, \qquad \boldsymbol{E} = \frac{1}{2}(\boldsymbol{F} + \boldsymbol{F}^T + \boldsymbol{F}^T\boldsymbol{F}),$$

where $\boldsymbol{E}$ is the Lagrangian Green strain tensor. The Young's Modulus E and the Poisson's ratio ν are defined as

$$E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu} \quad \text{and} \quad \nu = \frac{\lambda}{2(\lambda + \mu)}.$$

The Dirichlet and Neumann boundary conditions required to create a well-posed problem are a predefined deformation $\boldsymbol{u}_D$, and surface force $\boldsymbol{f}_N$,

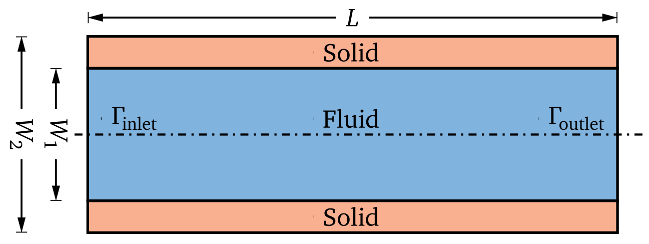
$$\boldsymbol{S} \cdot \boldsymbol{n} = \boldsymbol{f}_N \quad on \quad \Gamma_N, \qquad \text{and} \qquad \boldsymbol{u} = \boldsymbol{u}_D \quad on \quad \Gamma_D.$$

Coupling Conditions: In order to reproduce the monolithic solution for partitioned FSI coupling, a set of surface coupling conditions is required between the fluid and solid domains. The surface between the two solvers is defined as $\Gamma_{FS} = \Gamma_F \cap \Gamma_S$. The first coupling condition is the equality of displacements and velocities

$$\boldsymbol{v}_F = \frac{\delta \boldsymbol{u}_S}{\delta t} \qquad \text{on} \quad \Gamma_{FS},$$

which ensures that material from different physical fields do not overlap or separate. The second condition is the equality of forces

$$\boldsymbol{\sigma}_S \cdot \boldsymbol{n}_S = -\boldsymbol{\sigma}_F \cdot \boldsymbol{n}_F \qquad \text{on} \quad \Gamma_{FS}.$$



	Domain	
L	0.05	m
W_1	0.01	m
W_2	0.012	m
	Fluid	
ρ	1000	kg/m^3
ν	3×10^{-6}	$m^2 \times s^{-1}$
	Solid	
E	3×10^5	Pa
ρ	1200/120	kg/m^3

FIGURE 3.1 Domain geometry—not to scale—(left) and dimension and material parameters (right) of the Elastic-Tube-3D test case. The model introduced by [Ger03] represents a pressure wave travelling down an arterial tube. Two different solid material densities of 1200 kg/m^3 and 120 kg/m^3 are studied. The low density ratio between the fluid and solid results in a large added mass effect [För07]; [Van09].

The coupling scheme does not directly handle the coupling conditions, but rather the application of applying the coupling conditions in the relevant solvers only need to converge, i.e., the fixed-point equations are equivalent to the coupling conditions.

3.2.2 Test Cases

The experimental simulation test cases were selected to include standard test cases found in literature and extensions to represent more real-world problems and, thus, pose a larger challenge to the quasi-Newton coupling scheme. All tutorial cases can be found at: `https://github.com/KyleDavisSA/IQN-test-cases`.

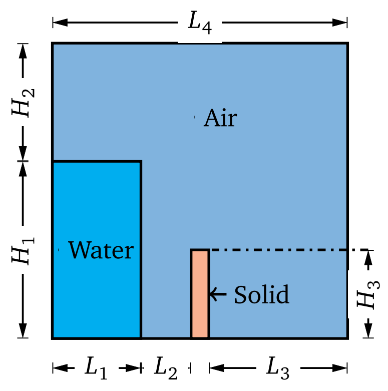
Elastic Tube 3D: The Elastic-Tube-3D problem, proposed by [Ger03] and used in numerous studies [Lin15]; [Spe20]; [Deg09], is a simplified haemodynamic FSI test case. The test case geometry consists of a cylindrical tube with an elastic wall and an inner fluid domain. The two domains exchange information along the length of the tube, such that the flow is constricted within the walls of the solid solver. A time-dependent pressure boundary condition is applied at the fluid inlet, along with a zero pressure boundary condition at the outlet. At the inlet, a pressure of 1.3332 kPa is applied for 3 ms, followed by 0 Pa for another 7 ms. At the outlet, the pressure is kept constant at 0 Pa. The simulation is run for a total of 10^{-2} s with a time step of $dt = 10^{-4}$ s for a total of 100 time steps. The domain geometry and the material properties are shown in Figure 3.1.

We compare two different densities for the structural solver to examine a case with strong added mass effect [För07]; [Van09]: (i) $\rho = 1200$ kg/m^3 and (ii) $\rho = 120$ kg/m^3, referred to as Elastic-Tube-3D-Heavy and Elastic-Tube-3D-Light, respectively. The low solid density of $\rho = 120$ kg/m^3 is almost an order of magnitude lower than the fluid density, adding a significant added mass effect to the problem. The structural solver has 11,736 elements in the domain and 1813 on the coupling interface, respectively. The fluid domain contains 32,691 finite volume cells, with

1860 vertices on the coupling interface.

As the simulation results in Section 3.3 present only the convergence and performance results of the enhanced quasi-Newton methods, the physical simulation results are briefly explained. A snapshot of the Elastic-Tube-3D-Heavy simulation at time $T = 40$ *ms* is shown in Figure 3.3 (A), where the fluid pressure and solid deformation are shown[4]. The left and the right boundary are fixed for the solid simulation and cannot move, while the rest of the tube is able to expand or contract. The initial pressure inlet boundary condition causes a pressure wave to form in the fluid and in the solid domain, which propagates down through the tube, causing a deforming "bulge" in the solid wall where the pressure is the largest. At $T = 40$ *ms* the inlet pressure is already switched to 0 *Pa* All software version to perform the simulations and visualise the results are found in Section 3.2.4.

Breaking-Dam-2D: The Breaking-Dam-2D test case is a free surface problem, where a large body of water comes into contact with a flexible barrier [Wal05]; [Bog16a]. This test case may pose problems for the quasi-Newton method: Firstly, the past information retained in the matrices $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ may not be entirely relevant once the water impacts the coupling interface and, thus, the character of the interaction between fluid and solid changes. Secondly, the pre-scaling weight values may change dramatically at the moment of the impact. The domain and the material properties are shown in Figure 3.2.

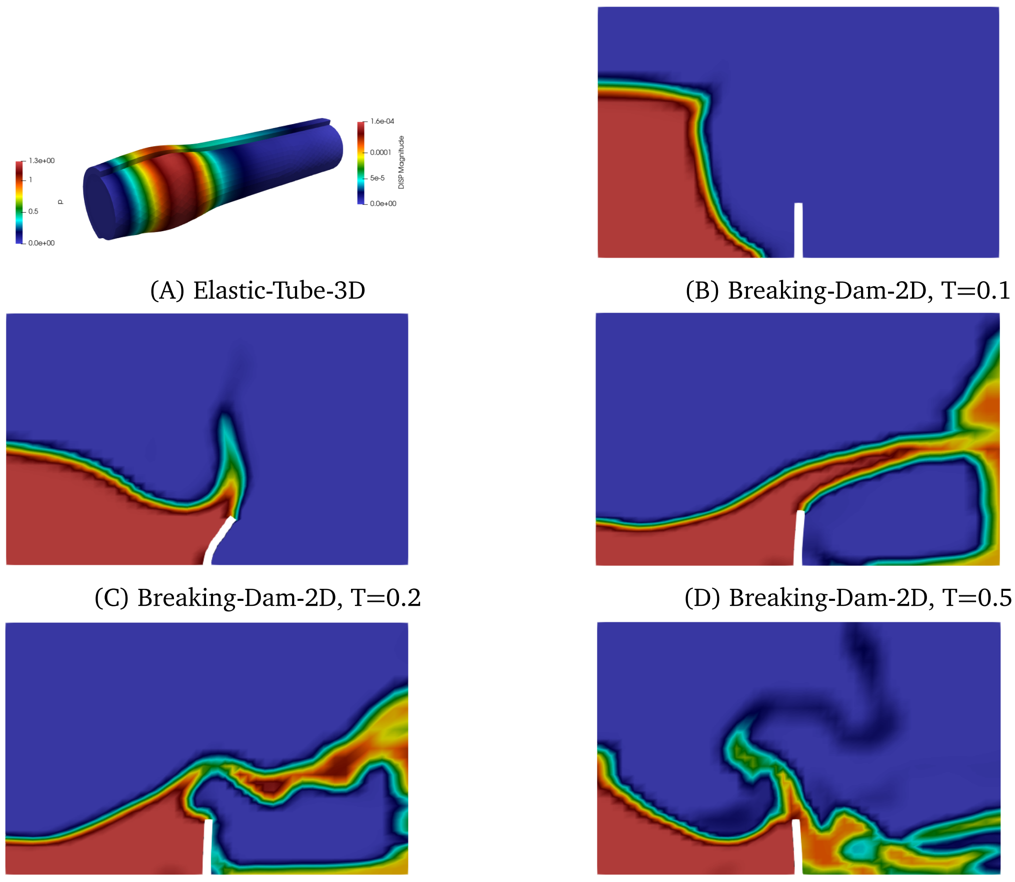


Domain					
L_1	0.146	m	H_1	0.292	m
L_2	0.14	m	H_2	0.073	m
L_3	0.286	m	H_3	0.08	m
L_4	0.548	m			
Fluid – Water			**Fluid – Air**		
ρ	1000	kg/m^3	ρ	1	kg/m^3
ν	1×10^{-6}	$m^2 \times s^{-1}$	ν	1×10^{-5}	$m^2 \times s^{-1}$
Solid					
E	3×10^5	Pa			
ρ	2500	kg/m^3			

FIGURE 3.2 Domain geometry—not to scale—(left) and dimension and material parameters (right) of the Breaking-Dam-2D test case. The test case models a body of water striking a solid wall, where the physical behaviour on the coupling interface changes when water is in contact with the wall. The case is described in [Wal05].

A no slip boundary condition is applied at the bottom, the left, and the right boundary, and a zero pressure condition at the top boundary. The test case was run for 1 *s* with a time step size of $dt = 0.005$ *s*, for a total of 200 time steps. The fluid domain contains 1382 cells in the domain, with 44 vertices on the interface, and the solid domain uses 325 finite elements in the domain, and 282 vertices on the coupling interface.

[4]the tube deformation has been amplified by a factor of 10.



(A) Elastic-Tube-3D

(B) Breaking-Dam-2D, T=0.1

(C) Breaking-Dam-2D, T=0.2

(D) Breaking-Dam-2D, T=0.5

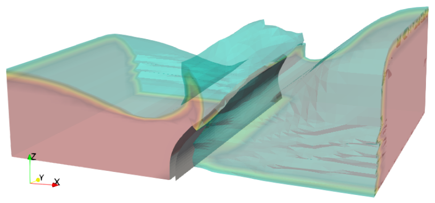
(E) Breaking-Dam-2D, T=0.6

(F) Breaking-Dam-2D, T=0.9

FIGURE 3.3 Simulation screenshots of the Elastic-Tube-3D and Breaking-Dam-2D test cases at various time steps. For the Elastic-Tube-3D, the displacement is amplified by a factor of 10, with a slice cut down the length of the solid domain tube. For the Breaking-Dam-2D, the air-water mixture is represented by the factor α, where $\alpha = 0$ represents air (blue) and $\alpha = 1$ represents water (red).

At the start of the simulation, the water column is static on the left, with gravity being the only force exerted on the water (Figure 3.3 (B)). Over time, the water column "falls" and moves towards the wall, eventually striking the wall and causing the wall to bend due to the force of the water (Figure 3.3 (C)). The water moves over the top of the wall (Figure 3.3 (D)), and eventually falls on both sides of the wall (Figure 3.3 (E) and Figure 3.3 (F)).

Breaking-Dam-3D: The Breaking-Dam-3D test case is a more complex test case inspired by the Breaking-Dam-2D example. A larger domain was created with bodies of water placed on either side of a solid wall. The water bodies are offset in the third dimension such that they hit the wall at opposite ends and at different times, resulting in a non-symmetric movement of the dam wall (Figure 3.4 (left)). The solid domain is fixed only at the bottom and the sides are free to move in-plane. The test case was run for 0.75 s with a time step of $dt = 0.005$ s, for a total of



Domain

L_1	2	m	H_2	2	m
L_2	0.95	m	H_3	1	m
L_3	1.95	m	H_4	0	m
L_4	1	m	H_5	4	m
L_5	6	m	H_6	1	m
H_1	2	m	H_7	4	m

Fluid – Water			**Fluid – Air**		
ρ	1000	$\mathrm{kg/m^3}$	ρ	1	$\mathrm{kg/m^3}$
ν	1×10^{-6}	$\mathrm{m}^2\times s^{-1}$	ν	1×10^{-5}	$\mathrm{m}^2\times s^{-1}$

Solid		
E	1×10^{7}	Pa
ρ	1000	$\mathrm{kg/m^3}$

FIGURE 3.4 Breaking-Dam-3D waters striking the flexible wall at 0.75 seconds (left) and dimensions and material parameters (right). The test case is an extension of the Breaking-Dam-2D model, that includes more complex 3D effects. The transparency of the water allows for the non-symmetric motion of the wall to be observed.

150 time steps. The domain geometry is shown in Figure 3.5, with the dimension and material properties given in Figure 3.4 (right). The fluid domain contains 25,712 cells in the domain with 714 vertices on the interface. The solid domain is simulated using 319 linear elements with 387 vertices on the coupling interface.

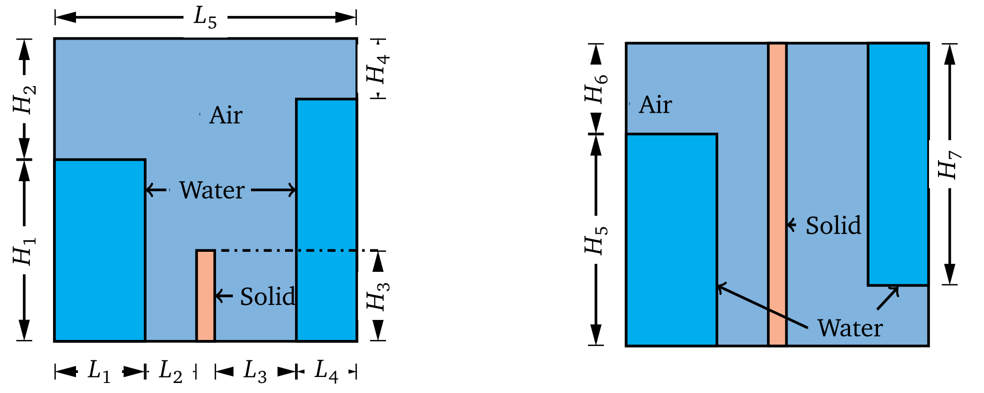
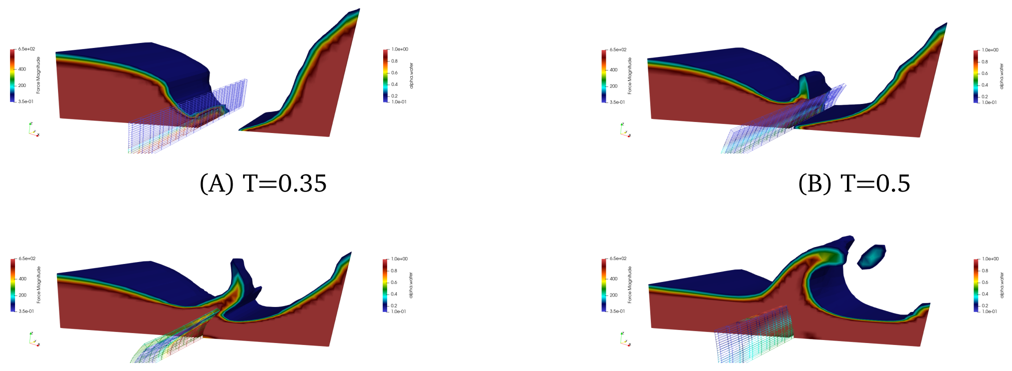


FIGURE 3.5 Domain geometry—not to scale— and dimensions of the Breaking-Dam-3D case with the front view (left) and top view (right). The test case is an extension of the Breaking-Dam-2D model, that includes more complex 3D effects of the water body and the solid structure.

3.2.3 Hardware

All simulations were performed on Neon cluster at the University of Stuttgart. The total computing resources available are: 4x Xeon E7-8880v3 (72 Core, 144 Threads), 512GB RAM, Ubuntu 20.04.


(A) T=0.35

(B) T=0.5

(C) T=0.6

(D) T=0.95

FIGURE 3.6 Snapshots of the Breaking-Dam-3D test case at various intervals. The water is shown striking the coupling interface (wireframe), where the force magnitude is plotted on the wireframe. The air-water mixture is represented by the factor α, where $\alpha = 0$ represent air (blue) and $\alpha = 1$ represent water (red). The left water column strikes the coupling interface first on the front half of the wall, followed by the right column on the back half of the wall (as seen from Figure 3.5 – right). This causes a twisting like motion of the wall during the deformation.

3.2.4 Software

All software used for the numerical experimental simulations is open-source and freely available. The following physics simulation software with versions numbers are:

- Fluid solver: `https://www.openfoam.com/news/main-news/openfoam-v20-12`, accessed on 07/10/2022, (OpenFOAM v2012) [Wel98];
- Fluid solver adapter: `https://github.com/precice/openfoam-adapter/releases/tag/v1.0.0`, accessed on 07/10/2022, OpenFOAM-preCICE Adapter v1.0.0;
- Solid solver: `http://www.calculix.de/`, accessed on 07/10/2022, CalculiX v2.17 [Dho04];
- Solid solver adapter: `https://github.com/precice/calculix-adapter/tree/5d42fb6160ede35926a59786ef8ae25dd71d7cdb`, accessed on CalculiX-preCICE Adapter, commit 5d42fb6.
- ParaView: v5.7.0 [Aya15].
- v2.3.0 as baseline without the enhancements:
  `https://github.com/precice/precice/releases/tag/v2.3.0`
- Branch iqnUpdates, commit 3fb3d8d: for the IQN-ILS enhancements presented in Section 3.1:
  `https://github.com/precice/precice/tree/3fb3d8d465e45e1eadba766a8ce5f1f96c138b20`
- Branch imvjUpdates, commit 9eeb5ca: for the IQN-IMVJ enhancements presented in Section 3.1:
  `https://github.com/precice/precice/tree/9eeb5ca50c1f1519e04fdf4a9ddf82c70795fbd0`

3.3 Numerical Experiments – Results and Discussion

In the current section, the numerical experiments testing the enhancements described in Section 3.1 are presented. The key components of the enhancements include: *(i)* the pre-scaling weight monitoring to reduce the number of QR-decompositions required, *(ii)* the new QR3 filter that mimics the operation of the QR2 filter, while being computational cheaper if combined with *(i)*, and *(iii)* the IMVJ modification, where more information can be added to $\boldsymbol{V}_k$ with low overhead in the QR-decomposition in combination with *(i)* and *(ii)*.

Only a limited amount of computing resources was required for testing the enhancements, as the new algorithms do not directly affect the parallel scalability of the algorithms. Computational savings are seen for both serial and parallel solver runs. Therefore, we do not focus on scaling measurements in this work but evaluate the stability and convergence of the new methods instead.

3.3.1 Pre-scaling Weight Monitoring

Purpose: By reducing the number of pre-scaling weight updates during the simulation runtime, it is possible to reduce the number of QR-decompositions performed. A simple way to reduce the number of pre-scaling weight updates is to simply freeze them, avoiding any further complete QR-recomputation. However, this may result in the pre-scaling weights being stuck at a sub-optimal value and selecting after which time step to freeze the weights adds another user defined parameter that may impact performance. Typically, the weights should be frozen after a few time steps to let any initial instabilities dissipate, while avoiding $\boldsymbol{V}_k$ from becoming too large. To solve the dilemma of when to freeze the weights, the pre-scaling weight monitoring was introduced in Section 3.1.1. The key performance criteria for evaluating the performance of the pre-scaling weight monitoring are: *(i)* the average number of iterations per time step should not increase when selecting pre-scaling monitoring and *(ii)*, the number of pre-scaling weight updates should be far lower than the total number of coupling iterations. In all test cases presented, the pre-scaling threshold of $\epsilon_{prescale} = 10$ in Equation 3.1. Therefore, the pre-scaling weights are updated if they change by an order of magnitude.

Results: The average number of iterations per time step for *(i)* the baseline simulation case (QR2 filter, pre-scaling weight update in each iteration), *(ii)* freezing the pre-scaling weights after 1, 2, 3 and 5 time steps, and *(iii)* pre-scaling weight monitoring, are compared in Table 3.2. The Breaking-Dam test cases used a filter limit of $\epsilon_{filter} = 0.1$, and the Elastic-Tube-3D test cases used a filter limit of $\epsilon_{filter} = 0.01$, as these settings resulted in the best performance for the respective case. Examining the average number of iterations per time steps provides an indication of the increase in computational cost compared to using a single physics solver only.

To help explain the deviation in performance for freezing the pre-scaling weights at different time steps, the ratio of the solid-to-fluid pre-scaling weights are shown in Figures 3.7, 3.8, 3.9, and 3.10. As the purpose of pre-scaling is to bring both fields within the same order of magnitude, the ratio between the fields is more important than their absolute magnitudes.

After considering the values of the pre-scaling weights chosen with different options and their impact on the number of iterations required, the next metric is the number of necessary pre-scaling weight update steps for the different strategies. Ideally, it is much smaller for the new monitoring strategy than for the old strategy where the weights are updated in each iteration. The average number of coupling iterations between pre-scaling weight updates is presented in Table 3.3, where larger values indicates that more iterations are performed before a pre-scaling update is performed, resulting in fewer complete QR-decompositions.

The following preCICE configuration values were used for all test cases:

- Data mapping: nearest-neighbor
- Acceleration: IQN-ILS
- Pre-scaling: residual-sum
- Max-iterations and max-time-steps-reused:
  - $\eta = 100$ and $\zeta = 10$ or
  - $\eta = 200$ and $\zeta = 20$.

Elastic-Tube-3D cases:

- Filter: QR2 filter, $\epsilon_{filter} = 0.01$

Breaking-Dam cases:

- Filter: QR2 filter, $\epsilon_{filter} = 0.1$

TABLE 3.2 Numerical results for pre-scaling weight monitoring: Average number of iterations per time step, comparing *(i)* standard QR-2 filter with residual-sum pre-scaling ('Base.'), *(ii)* freezing pre-scaling weights after time step 1 ('Freeze-1'), *(iii)* freezing after time step 2 ('Freeze-2'), *(iv)* freezing after time step 3 ('Freeze-3'), *(v)* freezing after time step 5 ('Freeze-5'), and *(vi)* using the pre-scaling monitoring ('Monitor.'). Column ζ is the number of time steps reused. The Breaking-Dam test cases used a filter limit of $\epsilon_{filter} = 0.1$, and the Elastic-Tube-3D test cases used a filter limit of $\epsilon_{filter} = 0.01$.

Test Case	ζ	**Base.**	**Freeze-1**	**Freeze-2**	**Freeze-3**	**Freeze-5**	**Monitor.**
Elastic-Tube-3D Heavy	10	4.51	4.45	4.40	4.47	4.65	4.48
Elastic-Tube-3D Heavy	20	3.84	3.96	3.87	4.10	3.95	3.94
Elastic-Tube-3D Light	10	7.23	7.30	7.07	7.30	7.20	7.16
Elastic-Tube-3D Light	20	5.84	6.00	5.83	5.97	5.95	5.83
Breaking-Dam-2D	10	4.14	4.66	3.85	6.79	6.01	4.09
Breaking-Dam-2D	20	4.12	4.80	4.05	7.18	6.55	4.15
Breaking-Dam-3D	10	5.64	6.02	5.77	5.62	5.69	5.65
Breaking-Dam-3D	20	5.42	5.78	5.39	5.42	5.48	5.44

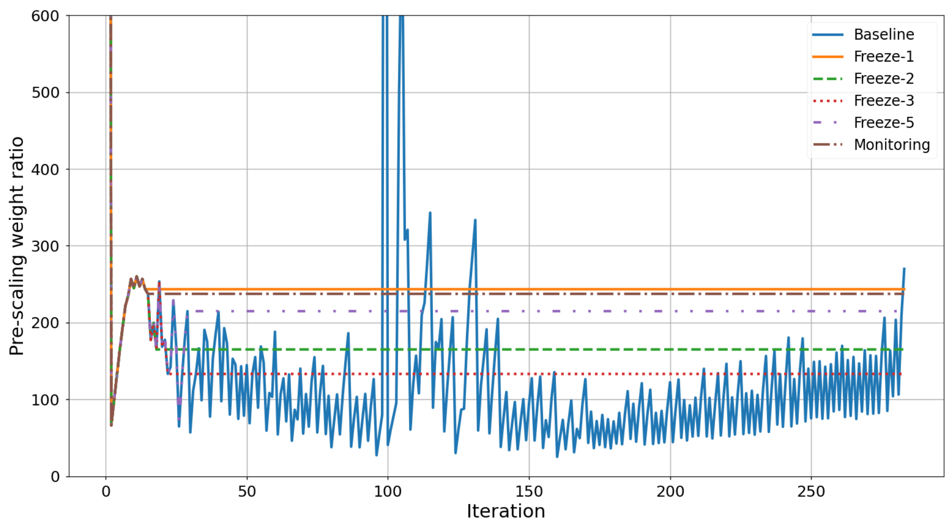


FIGURE 3.7 Numerical results for pre-scaling weight monitoring: Solid-to-fluid pre-scaling weight ratio for the Elastic-Tube-3D-Heavy test case. We show the weights as actually used for pre-scaling ($\bar{\lambda}_{k,i}$), not the theoretically computed values in each iteration ($\bar{\lambda}_{k,i}^{new}$).

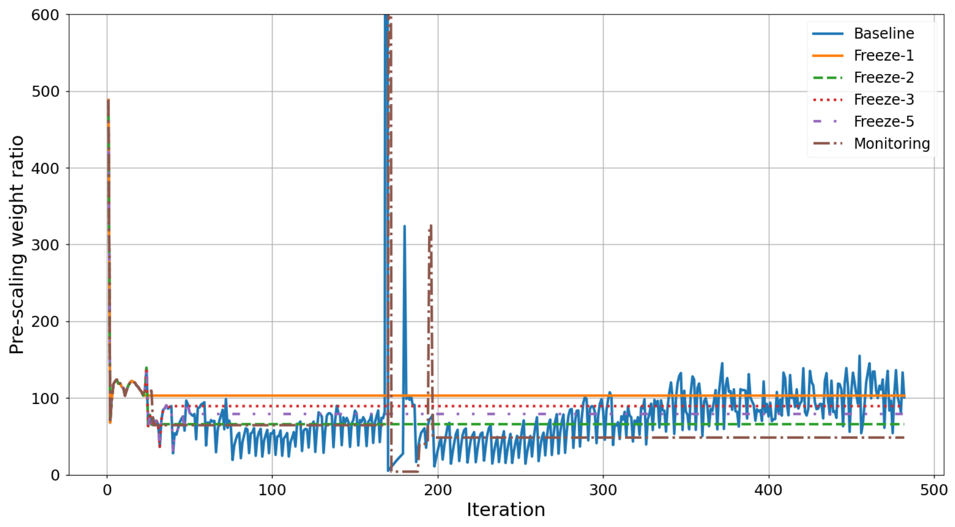


FIGURE 3.8 Numerical results for pre-scaling weight monitoring: Solid-to-fluid pre-scaling weight ratio for the Elastic-Tube-3D-Light test case. We show the weights as actually used for pre-scaling ($\bar{\lambda}_{k,i}$), not the theoretically computed values in each iteration ($\bar{\lambda}_{k,i}^{new}$).

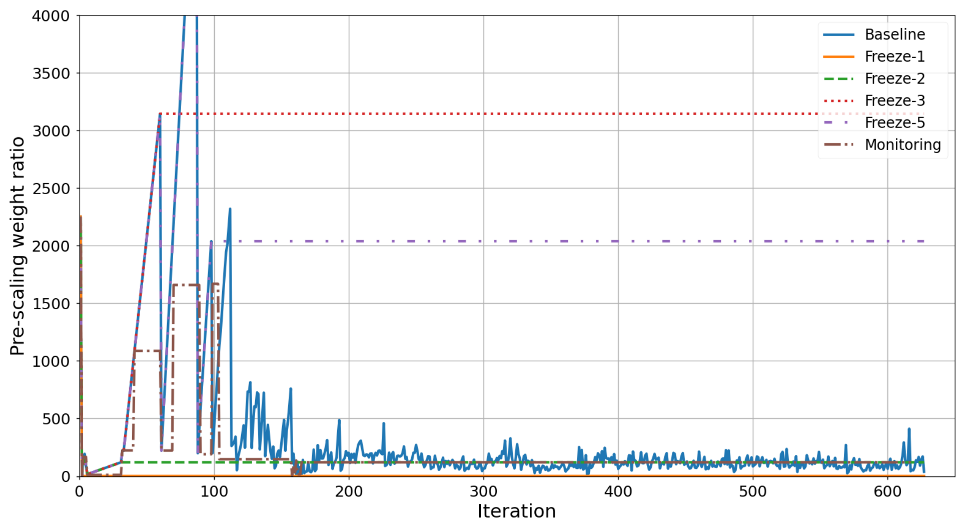


FIGURE 3.9 Numerical results for pre-scaling weight monitoring: Solid-to-fluid pre-scaling weight ratio for the Breaking-Dam-2D test case. We show the weights as actually used for pre-scaling ($\bar{\lambda}_{k,i}$), not the theoretically computed values in each iteration ($\bar{\lambda}_{k,i}^{new}$).

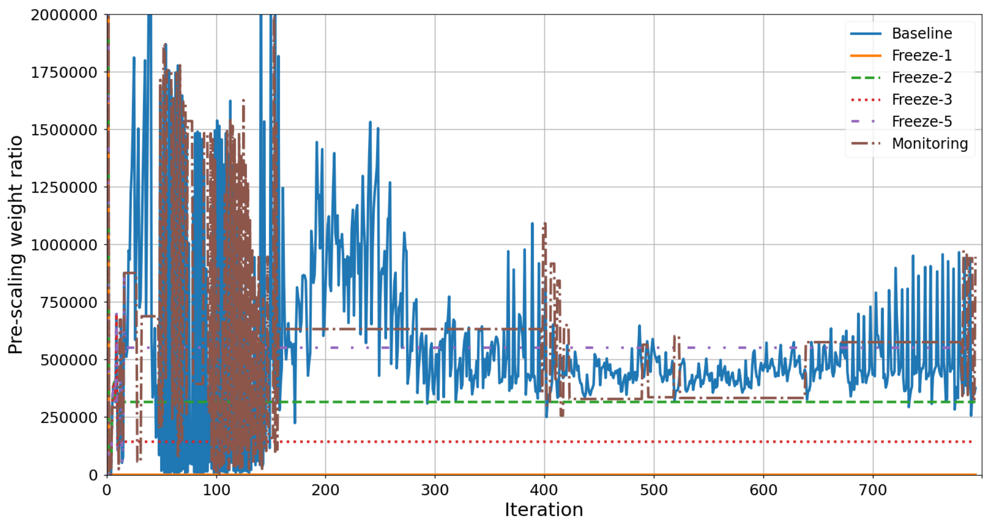


FIGURE 3.10 Numerical results for pre-scaling weight monitoring: Solid-to-fluid pre-scaling weight ratio for the Breaking-Dam-3D test case. We show the weights as actually used for pre-scaling ($\bar{\lambda}_{k,i}$), not the theoretically computed values in each iteration ($\bar{\lambda}_{k,i}^{new}$).

TABLE 3.3 Numerical results for pre-scaling weight monitoring: Average number of iterations between pre-scaling updates for the QR2 filter, comparing *(i)* residual-sum pre-scaling in every iterations ('Base.'), *(ii)* freezing pre-scaling weights after time step 1 ('Freeze-1'), *(iii)* freezing after time step 2 ('Freeze-2'), *(iv)* freezing after time step 3 ('Freeze-3'), *(v)* freezing after time step 5 ('Freeze-5'), and *(vi)* using the pre-scaling monitoring ('Monitor.').

Test Case	Base.	Freeze-1	Freeze-2	Freeze-3	Freeze-5	Monitor.
Elastic-Tube-3D Heavy	1.36	28.29	21.5	18.64	13.62	26.27
Elastic-Tube-3D Light	1.21	27.28	18.81	15.31	11.23	20.10
Breaking-Dam-2D	1.32	466	25.45	22.62	12.27	48.12
Breaking-Dam-3D	1.23	260.25	121.25	81.25	51.90	9.24

Discussion: When evaluating the results for the strategy of purely freezing the pre-scaling weights, the performance is highly dependent on the time step after which the weights are frozen, particularly for the Breaking-Dam test cases. Freezing the weights after the 2^{nd} time step appears to offer the best performance overall. For the Elastic-Tube-3D-Heavy test case, the number of iterations per time steps is not very sensitive to the time step when the weights are frozen. The same behaviour is also noticed for the Elastic-Tube-3D-Light test case, but with a larger number of iterations per time step due to the less dense solid structure. However, the Breaking-Dam-2D is much more sensitive to when the weights are frozen. Freezing the pre-scaling weights after 3 and 5 time steps performed significantly worse in this case. On the other hand, the pre-scaling weight monitoring performs well and remains close to the best result for each test case, but not necessarily optimally. Additionally, the pre-scaling weight monitoring performs better than the baseline case in 4 of the 8 test cases. This is not detrimental, as it is a parameterless default method that is able to adjust to any test case, while potentially offering significant computational savings.

We can conclude that the pre-scaling weight monitoring does not negatively impact the convergence rate of the coupled problems, while offering a parameter free method, significantly reducing the number of scaling weight updates and, therewith, the number of expensive complete QR decompositions.

The pre-scaling weight ratios when freezing at different time steps, for the Elastic-Tube-3D-Heavy case, are: $Freeze{-}1 = 1243.4$, $Freeze{-}2 = 165.0$, $Freeze{-}3 = 133.1$, $Freeze{-}5 = 214.8$. They are all very similar except for *Freeze-1*, which explains the fact that the number of iterations per time step does not vary significantly between the others. The Breaking-Dam-2D case exhibits the opposite behaviour, with vastly differing values for all freezing results: $Freeze - 1 = 1.002$, $Freeze - 2 = 120.32$, $Freeze - 3 = 3145.5$, $Freeze - 5 = 2039.13$.

Examining the baseline pre-scaling weights in Figures 3.7 and 3.8, the weights are quite stable, apart from the spike at iteration 100 and 175 for each figure respectively. This spike is due to the sudden change of the pressure inlet boundary condition reducing to 0 Pa after the

30th time step. The pre-scaling weights are scaled according to the residual $\boldsymbol{R}(\boldsymbol{x})$, and therefore a sudden, but short-lived change in the residual behaviour is seen.

Even though this spike quickly stabilises, it is not possible to know how the residuals will respond to a change in boundary conditions. This is highlighted during the initial unstable phase in the Breaking-Dam-2D and Breaking-Dam-3D test cases in Figure 3.9 and Figure 3.10. However, these unstable periods stabilise after 150 and 300 iterations, respectively. In these cases, freezing the weights during this initial phase results in extreme values for the pre-scaling weights. An important note is that the simulation itself is not unstable, it is only the difference in solver residuals that fluctuates significantly between time steps. This is assumed to be due to the relatively stiff solid wall interacting with air. The residual of the forces can fluctuate significantly, while applying a very small amount of pressure on the solid and causing almost no change in deformation. Even though the force values fluctuate and therefore the pre-scaling fluctuates, the convergence threshold of preCICE is still met as these fluctuating values are very small. As expected, the pre-scaling weight monitoring method has long periods of constant pre-scaling weights, followed by quick changes where required. This often occurs for the first iteration during a time step, where the residual values and, thus, the pre-scaling weights are the largest, and subsequently, smoothly decreasing due to the summation in equation 2.28.

The pre-scaling weight monitoring has the potential to reduce the computational cost of the QR-decomposition only if there are fewer pre-scaling weight updates compared to the baseline scenario. Examining Table 3.3, we see that the new monitoring method is able to provide suitable pre-scaling between the sub-vectors with a fraction of the number of updates compared to the baseline scenario. The average value for the baseline case with updates in every iteration in principle is slightly larger than one (as opposed to being $= 1$). The slight deviation from 1 is due to the fact that the weights are not updated during the last iteration of each time step, but instead reset to zero. The performance decreases for all cases from $Freeze-1$ to $Freeze-5$, in line with the additional pre-scaling weight updates required up to the 5th time step. The pre-scaling weight monitoring performs excellently, falling between $Freeze-1$ and $Freeze-2$ for all cases except the Breaking-Dam-3D. However, the pre-scaling weight monitoring is still approximately $8\times$ better than the baseline simulation case. Therefore, the pre-scaling weight monitoring does not increase the number of coupling iterations, while in the worst case from the tests provided, requires an order of magnitude fewer complete QR-decompositions.

3.3.2 Filtering

Purpose: The numerical experiments for the new pre-scaling weight monitoring in Section 3.3.1 confirmed that the pre-scaling weight monitoring successfully reduced the number of QR-decompositions that would be performed by reducing the number of pre-scaling weight updates required, while simultaneously keeping the average number of coupling iterations per time step similar to the baseline case. As the QR2 filter was selected, a complete QR-decomposition was performed due to the choice of filter. In the following experiments, the new QR3 filter that mimics the function of the QR2 filter, but with a drastically lower computational cost, is tested. The key criterion for evaluating the performance of the QR3 filter are: *(i)* the average number of iterations per time step must not increase (should remain similar to the number of iterations when using the QR2 filter) and *(ii)* the runtime reduction resulting from a low percentage of iterations that require a complete QR-decomposition. As the QR decomposition is only expensive once the size of $\boldsymbol{V}_k$ grows too large, we limit the testing of the filter to the IQN-ILS method. In addition to testing the QR3 filter, the influence of the number of time steps reused is simultaneously evaluated to find "good" default values without the need for parameter tuning.

The following preCICE configuration values were used for all test cases:

- Data mapping: nearest-neighbor
- Acceleration: IQN-ILS
- Pre-scaling: residual-sum with pre-scaling weight monitoring
- Max-iterations: $\eta = 1000$
- Filter: QR2 and QR3
- Elastic-Tube – filter limit: $\epsilon = 0.1, 0.01, 0.001$
- Breaking-Dam – filter limit: $\epsilon = 0.1, 0.01$[a]

[a]Breaking-Dam cases diverge for small filter limit values.

Results:

The average number of iterations per time step for all test cases are shown in Table 3.4. The number of time steps reused was set at set intervals of ζ =10, 20, 40, 60 and 100. The maximum number of iterations reused was set to $\eta = 1000$. Therefore, only ζ controls the size of $\boldsymbol{V}_k$ and $\boldsymbol{W}_k$ as the limit $\eta = 1000$ should not be met. The Breaking-Dam cases were not evaluated at $\epsilon = 0.001$ as the cases diverged. To provide a fair comparison, both the QR2 and QR3 filters use the pre-scaling weight monitoring, such that the pre-scaling weights do not influence the results.

The average number of coupling iterations that occurs for each column removed, for both QR2 and the QR3 filter, is shown in Table 3.4. A complete QR-decomposition is performed in each iteration for the QR2 filter step, or each time a column is tagged for deletion by the QR3 filter (which then simply performs the QR2 filter step). As more than 1 column might be removed by the QR2 filter step, Table 3.5 indicates maximum number of complete QR-decompositions performed by the QR3 filter. Larger values in the table translate to fewer column deletions during simulation runtime, and therefore fewer QR-decompositions performed due to filtering.

The QR filtering time as a percentage of the total simulation runtime, for both QR2 and QR3 filter, are shown in Table 3.6. The total simulation time includes running the solvers and preCICE. For the QR3 filter, the filter time includes the conditional check if columns are to be tagged for removal, and the QR2 filter if a column is tagged for removal.

Discussion:

The aim of the QR3 filter is to reduce the average cost per iteration with an optimised filtering

TABLE 3.4 Numerical experiments for the new QR3 filter: Average number of iterations per time step for the QR2 and QR3 filter for all test cases. ζ represents the number of time steps reused, with the maximum number of iterations reused $\eta = 1000$, such that ζ is the only variable controlling the size of $\boldsymbol{V}_k$. Both QR2 and QR3 filters use the pre-scaling weight monitoring to provide an accurate comparison.

			QR2			QR3	
Test Case	ζ	*0.001*	*0.01*	*0.1*	*0.001*	*0.01*	*0.1*
Elastic-Tube-3D-Heavy	10	4.26	4.51	5.12	4.59	4.48	5.24
Elastic-Tube-3D-Heavy	20	4.09	3.84	4.91	4.05	3.94	3.92
Elastic-Tube-3D-Heavy	40	3.75	3.69	4	3.77	3.32	3.57
Elastic-Tube-3D-Heavy	60	3.72	3.59	3.95	3.6	3.33	3.51
Elastic-Tube-3D-Heavy	100	3.71	3.57	3.93	3.57	3.23	3.44
Elastic-Tube-3D-Light	10	7.27	7.23	8.83	7.18	7.16	8.69
Elastic-Tube-3D-Light	20	5.78	5.84	7.88	5.83	5.83	7.67
Elastic-Tube-3D-Light	40	4.83	4.85	7	4.85	4.96	5.77
Elastic-Tube-3D-Light	60	4.55	4.57	6.57	4.61	4.68	5.43
Elastic-Tube-3D-Light	100	4.18	4.19	6.21	4.27	4.27	5.03
Breaking-Dam-2D	10	—	4.185	4.14	—	3.99	3.905
Breaking-Dam-2D	20	—	4.455	4.21	—	4.44	4
Breaking-Dam-2D	40	—	4.625	4.375	—	4.585	4.175
Breaking-Dam-2D	60	—	4.765	4.375	—	4.665	4.175
Breaking-Dam-2D	100	—	4.755	4.375	—	4.665	4.175
Breaking-Dam-3D	10	—	5.9	5.66	—	5.68	5.69
Breaking-Dam-3D	20	—	5.66	5.42	—	5.94	5.48
Breaking-Dam-3D	40	—	7.33	5.77	—	7.37	5.69
Breaking-Dam-3D	60	—	8.01	6.02	—	8.46	5.97
Breaking-Dam-3D	100	—	8.58	6.15	—	8.64	6.049

algorithm. However, care must be taken that this does not increase the number of quasi-Newton iterations substantially. Even though the QR2 filter results in fewer iterations per time step for 67 of the 100 simulations in Table 3.4, the results are always similar for the QR3 filter.

Examining Table 3.4, the number of coupling iterations decreases for both Elastic-Tube-3D test cases when increasing the number of time steps reused. However, the opposite is noticed for the Breaking-Dam test cases, where the number of coupling iterations increase for increasing ζ. In the Elastic-Tube-3D-Heavy test case with QR3 filter and $\epsilon_{filter} = 0.1$, the number of iterations per time step reduces from 5.24 to 3.57 between $\zeta = 10$ and $\zeta = 40$. However, only a small reduction is observed when increasing from $\zeta = 40$ to $\zeta = 100$, with 3.57 and 3.44 iterations per time step, respectively. Similar behaviour is observed for the QR2 filter, and for the Elastic-Tube-3D-Light test case for both the QR2 and the QR3 filter. Therefore, the increase in performance (reduction in number of iterations per time step) starts to stagnate for number of time steps reused greater than $\zeta = 40$. Only a moderate performance gain is seen after that.

The complete opposite behaviour was found for the Breaking-Dam test cases. As the number of time steps reused increases, the performance of the degrades. For the Breaking-Dam-2D test

TABLE 3.5 Numerical experiments for the new QR3 filter: Average number of iterations per column deleted for the QR2 and the QR3 filter. The larger the value, the more coupling iterations are performed between each column that is removed during the filter step. Larger values in the table translates to fewer columns deletions during simulation runtime, and therefore fewer QR-decompositions performed by the QR3 filter. As more than 1 column can be deleted during a single filter step, this represents a worst-case performance for the QR3 filter. The QR2 filter always performs a QR-decomposition in each iteration.

			QR2			**QR3**	
Test Case	ζ	*0.001*	*0.01*	*0.1*	*0.001*	*0.01*	*0.1*
Elastic-Tube-3D-Heavy	10	426	45.1	2.55	459	149.33	3.25
Elastic-Tube-3D-Heavy	20	409	96	2.24	405	131.33	12.65
Elastic-Tube-3D-Heavy	40	125	28.38	2.99	377	110.67	11.9
Elastic-Tube-3D-Heavy	60	124	29.92	2.95	120	55.5	10.32
Elastic-Tube-3D-Heavy	100	123.67	29.75	2.91	119	53.83	9.83
Elastic-Tube-3D-Light	10	727	180.75	3.31	718	716	3.02
Elastic-Tube-3D-Light	20	578	194.67	2.81	583	194.33	2.89
Elastic-Tube-3D-Light	40	483	69.29	2.47	485	124	4.47
Elastic-Tube-3D-Light	60	455	65.29	2.43	461	117	4.08
Elastic-Tube-3D-Light	100	418	59.86	2.20	427	85.40	3.84
Breaking-Dam-2D	10	–	4.5	1.86	–	5.91	1.97
Breaking-Dam-2D	20	–	2.06	1.44	–	2.06	1.49
Breaking-Dam-2D	40	–	1.47	1.38	–	1.54	1.39
Breaking-Dam-2D	60	–	1.45	1.38	–	1.42	1.39
Breaking-Dam-2D	100	–	1.45	1.38	–	1.42	1.39
Breaking-Dam-3D	10	–	12.95	6.40	–	16.77	8.82
Breaking-Dam-3D	20	–	9.25	4.53	–	8.23	6.11
Breaking-Dam-3D	40	–	6.08	2.71	–	6.09	3.29
Breaking-Dam-3D	60	–	3.47	2.20	–	2.69	2.32
Breaking-Dam-3D	100	–	2.84	1.97	–	2.46	2.13

case with the QR3 filter and $\epsilon_{filter} = 0.01$, the number of iterations per time step increases from 3.99 to 4.665 when $\zeta = 10$ increases to $\zeta = 100$. This is less severe than for the Breaking-Dam-3D test case (QR3 filter and $\epsilon_{filter} = 0.01$), where the number of iterations per time step increased from 5.68 to 8.64 when $\zeta = 10$ increases to $\zeta = 100$.

For both Breaking-Dam test cases, the filter limit $\epsilon_{filter} = 0.1$ results in fewer iterations per time step, whereas $\epsilon_{filter} = 0.01$ performed better for the Elastic-Tube-3D test cases. For the Breaking-Dam test cases, there could be a large build-up of linearly dependent columns and the large filter limit required deletes too many columns. Therefore, $\boldsymbol{V}_k$ does not contain a large enough history to speed up the convergence. Or alternatively, the very old information does not provide a suitable search space for the quasi-Newton method to solve the fixed-point equation. By selecting these test cases, it is clear that the parameter selection is non-trivial and highly dependent on the nature of the problem to solve. However, even for the Breaking-Dam test cases, going below $\zeta = 10$ is not recommended as the amount of previous information is very limited for too low numbers of reused time steps.

TABLE 3.6 Numerical experiments for the new QR3 filter: QR filter time as percentage of the entire simulation runtime for the QR2 and QR3 filters. The QR3 filter time includes the time to perform the check if a column is to be deleted, and the QR2 filter step if required. The measurement does not include the initial column insertion process into Q and R.

		QR2			**QR3**		
Test Case	ζ	*0.001*	*0.01*	*0.1*	*0.001*	*0.01*	*0.1*
Elastic-Tube-3D	10	2.59%	3.27%	1.99%	0.062%	0.07%	0.78%
Elastic-Tube-3D	20	8.62%	4.67%	4.04%	0.07%	0.08%	0.43%
Elastic-Tube-3D	40	15.81%	14.29%	8.34%	0.19%	0.11%	0.95%
Elastic-Tube-3D	60	24.35%	20.99%	10.43%	0.27%	0.38%	1.75%
Elastic-Tube-3D	100	31.69%	27.41%	12.54%	0.28%	0.41%	2.16%
Elastic-Tube-3D-Light	10	8.98%	8.72%	7.90%	0.14%	0.15%	2.80%
Elastic-Tube-3D-Light	20	13.15%	10.90%	14.58%	0.39%	0.44%	5.61%
Elastic-Tube-3D-Light	40	23.43%	23.59%	24.25%	0.59%	0.60%	5.89%
Elastic-Tube-3D-Light	60	31.92%	31.40%	28.66%	0.66%	0.68%	8.26%
Elastic-Tube-3D-Light	100	39.52%	37.79%	33.40%	0.71%	1.11%	9.76%
Breaking-Dam-2D	10	–	1.12%	0.65%	–	0.18%	0.36%
Breaking-Dam-2D	20	–	3.06%	1.09%	–	1.56%	0.75%
Breaking-Dam-2D	40	–	5.28%	1.53%	–	4.01%	0.98%
Breaking-Dam-2D	60	–	7.12%	1.56%	–	4.89%	0.98%
Breaking-Dam-2D	100	–	7.21%	1.55%	–	4.98%	0.99%
Breaking-Dam-3D	10	–	2.34%	2.24%	–	0.092%	0.22%
Breaking-Dam-3D	20	–	5.58%	5.23%	–	0.31%	0.63%
Breaking-Dam-3D	40	–	24.78%	13.82%	–	2.35%	4.87%
Breaking-Dam-3D	60	–	37.50%	18.01%	–	14.54%	9.86%
Breaking-Dam-3D	100	–	50.15%	21.70%	–	18.17%	12.50%

Next, we examine the number of iterations per column deleted in Table 3.5. The total number of iterations is divided by number of columns deleted throughout the entire simulation. As multiple columns can be removed during a single QR filter step, this represents a worst-case scenario for the number of complete QR-decompositions for the QR3 filter.

For the Elastic-Tube-3D cases with $\epsilon_{filter} = 0.001$, the number of iterations per column deleted equals the total number of iterations for the whole simulation. On further inspection, only the very first column was deleted during the first time step. This column was generated from the under-relaxation step in the first iteration of the simulation. Afterwards, no more columns were deleted as the filter limit threshold was too low. All test cases show that filtering reduces the number of iterations per time step, and the only question is to what extent the filter must be applied, i.e., which filter limit is optimal. The largest filter limit of $\epsilon_{filter} = 0.1$ results in more columns being deleted. However, fewer columns are deleted for the QR3 filter, resulting in fewer complete QR-decompositions. Regardless of the filter limit, the Breaking-Dam-2D always has many columns deleted, between 1.38 and 2.06 iterations per column removed for all input settings apart from $\epsilon_{filter} = 0.01$ and $\zeta = 10$. Therefore, there is little expected gain in runtime from the QR3 filter for this case.

Finally, the impact on the computational runtime is evaluated in Table 3.6. Examining the QR2 filter, increasing the number of time steps reused ζ, the filtering time starts to play a larger role in the simulation runtime. This is especially significant for the Elastic-Tube-3D cases with $\epsilon_{filter} = 0.001$, where the filtering time increase from 2.59% to 31.69% for the Elastic-Tube-3D-Heavy, and from 8.98% to 39.52% for the Elastic-Tube-3D-Light. For the Breaking-Dam-3D test case with $\epsilon_{filter} = 0.01$, up to half of the simulation runtime is due to filtering.

The QR3 filter results in a dramatic reduction in filtering runtime. The percentage filtering time reduced from 31.69% to 0.28%, and from 39.52% to 0.71% for the Elastic-Tube-3D-Heavy and Elastic-Tube-3D-Light, respectively. At larger ζ values for the Breaking-Dam cases, increasing the filter limit from $\epsilon_{filter} = 0.01$ to $\epsilon_{filter} = 0.1$ with QR3 filter results in a reduction in filtering time, opposite to the behaviour of the Elastic-Tube-3D cases where the filtering time increases.

The Elastic-Tube-3D cases with the QR3 filter results in fewer columns removed, and combined with the cheaper filtering procedure, significant savings in computational runtime are observed. The savings in runtime with the QR3 filter are up to $1/3^{rd}$ of the simulation runtime for some cases. For small ζ values, i.e., few reused time steps, the filtering time decreases by almost 2 orders of magnitude. However, the biggest gains to notice are for $\epsilon_{filter} = 0.001$, as the QR-decomposition step is almost never triggered with the QR3 filter even for large ζ, and the filtering time is made up almost entirely of column insertion steps only. The improvement is less pronounced for $\epsilon_{filter} = 0.1$, as the high threshold often triggers a QR2 filter step, yet the runtime savings are still significant.

The Breaking-Dam test cases showed milder improvements, as many QR2 filter steps are performed even with the QR3 filter with a low threshold value. However, any reduction in QR2 filter steps already yields a noticeable improvement in runtime.

3.3.3 IQN-IMVJ Modification

Purpose: With the introduction of the pre-scaling weight monitoring and the QR3 filter, the size of $\boldsymbol{V}_k$, and associated cost of the QR-decomposition, are no longer a bottleneck preventing fast runtimes. With this, it is possible to examine if the IQN-IMVJ method is also able to benefit from this cost saving. The inverse Jacobian is never explicitly determined, but an on-the-fly approximation as the sum of low-rank products of $\boldsymbol{V}_k$ and $\widetilde{\boldsymbol{W}}_k$ matrices is used instead, If larger intervals between updates of $\boldsymbol{J}_{prev}^{-1}$ and, accordingly, keeping more columns in the current $\boldsymbol{V}_k$ and $\widetilde{\boldsymbol{W}}_k$, more information is accessible to filtering and the probability for the QR filter to remove columns increases. Therefore, we can potentially remove columns that are harmful to the convergence of the coupled simulation. In this section, we evaluate if concatenating columns from multiple time steps, and filtering $\boldsymbol{V}_k$, reduces the number of iterations per time step. The key criteria for the evaluating the performance are: *(i)* the average number of iterations per time step should decrease as more columns are placed in the current $\boldsymbol{V}_k$, *(ii)* the computational cost of building $\widetilde{\boldsymbol{W}}_k$ and performing the restart should not increase.

The following preCICE configuration values were used for all test cases:

- Data mapping: nearest-neighbor
- Acceleration: IQN-IMVJ
- Pre-scaling: residual-sum with pre-scaling weight monitoring
- Max-iterations: $\eta = 1000$
- Max-time-steps-reused: $\zeta = 10, 20, 40, 60, 100$
- Filter: QR2 and QR3
- Elastic-Tube-3D – filter limit: $\epsilon_{filter} = 0.01$
- Breaking-Dam – filter limit: $\epsilon_{filter} = 0.1$

Results: The average number of iterations per time step versus the number of time steps reused per group (i.e., number of time steps between updates of $\boldsymbol{J}_{prev}^{-1}$ and retained in the current pair $\boldsymbol{V}_k$ and $\widetilde{\boldsymbol{W}}_k$) are shown in Figures 3.11 and 3.12 for the Elastic-Tube-3D-Heavy and the Elastic-Tube-3D-Light test cases, respectively, and Figures 3.13 and 3.14 for the Breaking-Dam-2D and the Breaking-Dam-3D test cases, respectively. In each figure, the total number of time steps performed before a restart, is shown from left to the right for a restart every 10 time steps (left, Restart-10), 20 time steps (middle, Restart-20) and 40 time steps (right, Restart-40), respectively. In each graph, the x-axis displays the number of time steps that are placed into each group. Therefore, the maximum number of time steps reused per group is bounded by the number of time steps per restart. If the time steps reused is equal to the time steps between restarts, then a restart is performed each time a new group is created.

The computational runtime of the algorithmic block `Apply Filter`, `Restart Time` and `Build Wtil`, for the QR2 and QR3 filter while varying time steps per group and between restarts, is shown in Figure 3.15. The `Apply Filter` is the time to perform the QR2 or QR3 filtering step. The `Restart Time` is the computational time to generate $\widetilde{\boldsymbol{W}}_{k_{res}}$ and $\boldsymbol{V}_{k_{res}}^{\dagger}$ from Equation 2.25 (also defined as steps "RS-SVD Update" or "RS-LS Update" in Table 3.1). The `Build Wtil` is the time to rebuild $\widetilde{\boldsymbol{W}}_{k_{res}}$ from $\boldsymbol{W}_{k_{res}}$ once per time step (the "Update $\tilde{\boldsymbol{w}}$" step performed η times in Table 3.1).

Discussion: The influence that the number of time steps reused per group has on the number of iterations per time step is evaluated, for varying number of time steps between restarts. Examining the Elastic-Tube-3D-Heavy (Figure 3.11) test cases with the RS-LS restart procedure, using *5* time steps per group always improves the performance of the IMVJ method, apart for 20 time steps between restarts. However, the RS-SVD restart method always has a reduction in iterations per time step when using *2* and *5* time steps per group.

A noticeable spike in the Elastic-Tube-3D-Heavy is observed for using *10* time steps per group at Restart-20 for RS-SVD, but the number of iterations drops when using *20* time steps per group. The number of iterations reduces for all number of time steps per group for Restart-40, even for

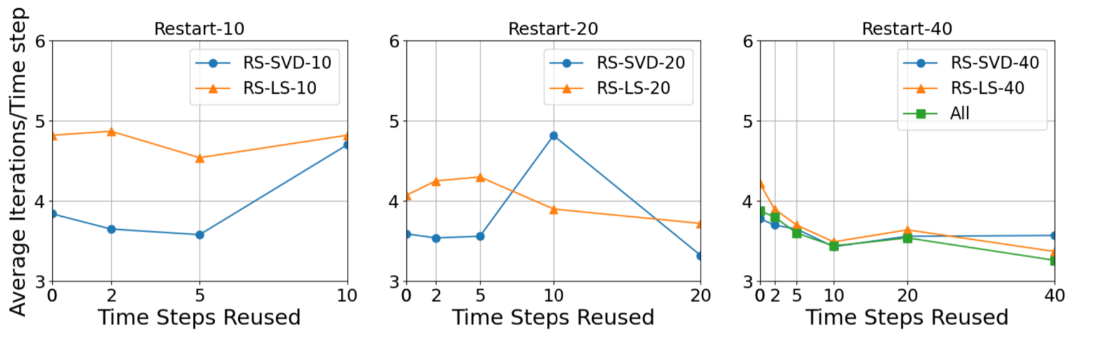


FIGURE 3.11 Numerical experiments for the IQN-IMVJ modification: Average number of iterations vs. time step reused per group for the Elastic-Tube-3D-Heavy test case for three restart scenarios with IQN-IMVJ restart every 10 (left), 20 (middle), and 40 (right) time steps. The maximum number of time steps reused is bounded by the number of time steps between restarts.

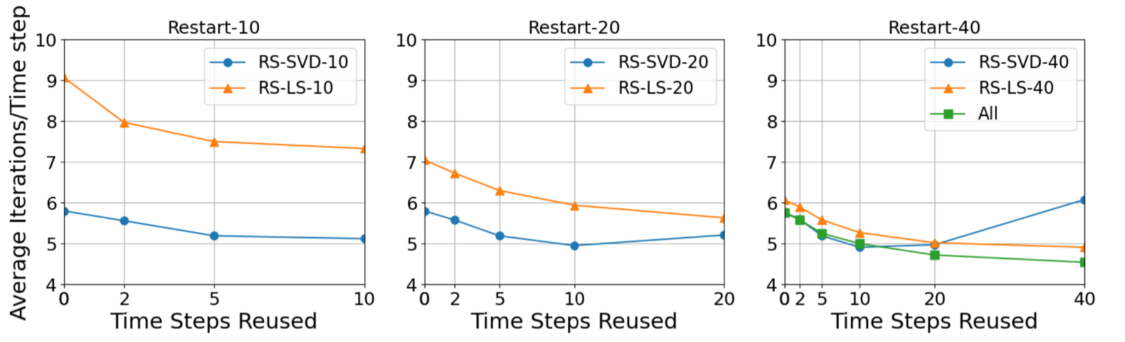


FIGURE 3.12 Numerical experiments for the IQN-IMVJ modification: Average number of iterations vs. time step reused per group for the Elastic-Tube-3D-Light test case for three restart scenarios with IQN-IMVJ restart every 10 (left), 20 (middle), and 40 (right) time steps. The maximum number of time steps reused is bounded by the number of time steps between restarts.

the *All* case, where no restart is performed, and all groups throughout the simulation runtime are kept in memory for the on-the-fly computation.

The IQN-IMVJ information reuse reduced the number of coupling iterations for all input configurations in the Elastic-Tube-3D-Light (Figure 3.12) test cases, apart for the RS-SVD with *40* time steps per chunk. In almost opposite behaviour to the Elastic-Tube-3D-Heavy case, the Restart-10 using *10* time steps per group does not result in an increased number of iterations.

For the Breaking-Dam-2D (Figure 3.13) cases, a reduction in coupling iterations is observed when using more than 1 time step per group. However, if too many time steps are concatenated into one group, the performance starts to degrade again, similar to the behaviour of the IQN-

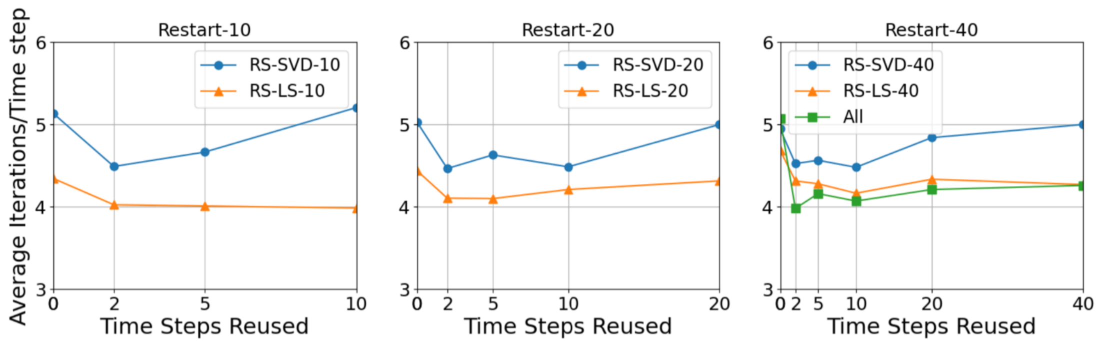


FIGURE 3.13 Numerical experiments for the IQN-IMVJ modification: Average number of iterations vs. time step reused per group for the Breaking-Dam-2D test case for three restart scenarios with IQN-IMVJ restart every 10 (left), 20 (middle), and 40 (right) time steps. The maximum number of time steps reused is bounded by the number of time steps between restarts.

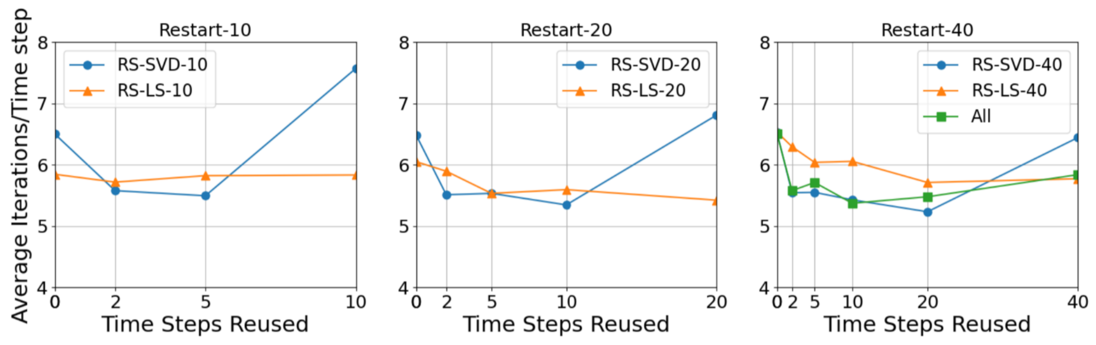


FIGURE 3.14 Numerical experiments for the IQN-IMVJ modification: Average number of iterations vs. time step reused per group for the Breaking-Dam-3D test case for three restart scenarios with IQN-IMVJ restart every 10 (left), 20 (middle), and 40 (right) time steps. The maximum number of time steps reused is upper bound by the number of time steps between restarts.

ILS method. The RS-LS restart method always outperforms the RS-SVD restart method. Not performing a restart, results in fewer iterations for the Restart-40 configuration. However, Restart-10 is the best performing configuration set.

Similar to the case above, the Breaking-Dam-3D (Figure 3.14) has an initial reduction in iterations when increasing the number of time steps per group. However, the performance degrades again as more time steps are placed into a single chunk. For the Restart-40 configuration, the RS-SVD and the *All* configuration outperforms the RS-LS method, which is opposite to the Breaking-Dam-2D case.

For both Elastic-Tube-3D test cases, the RS-SVD restart method outperforms the RS-LS

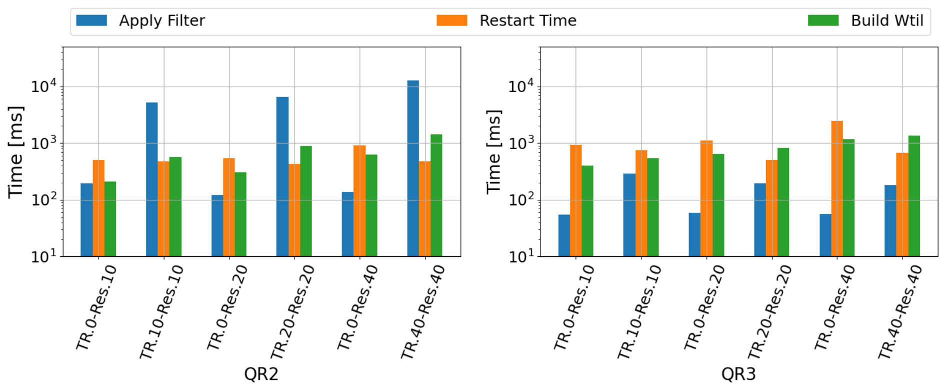


FIGURE 3.15 Numerical experiments for the IQN-IMVJ modification: the computational runtime of *Apply Filter*, *Restart Time* and *Build Wtil* algorithms for the QR2 and the QR3 filter with varying time steps per group and between restarts. The x-axis indicates the number of time steps per group (*TR.0* indicates 0 time steps reused per group, and *TR.10* reuses 10 time steps per group) and number of time steps between restarts (*Res.10* indicates that 10 time steps are performed between restarts).

method. This is somewhat expected, as the IQN-ILS performs better for these cases with larger values of ζ and the RS-SVD can approximate the inverse Jacobian over the entire simulation runtime. However, when restarting after 40 time steps, there is little difference between RS-SVD, RS-LS and *All*, probably as enough information is contained within the first 40 time steps for the RS-LS to compete with the other methods. In both Figures 3.11 and 3.12, using at least *2* or *5* time steps per group reduces the number of coupling iterations. However, the performance of the Elastic-Tube-3D-Heavy case does not show consistent behaviour at *10* time steps per group, unless using 40 time steps per restart.

The inconsistent behaviour is difficult to diagnose for the Breaking-Dam cases, as the performance degrades with more time steps per group. A potential cause is that, when a new group is created, the matrices are empty. Therefore, the information contained in the current $\boldsymbol{V}_k$ and $\widetilde{\boldsymbol{W}}_k$ matrices used in the multi-secant equation needs to be fulfilled exactly, whereas all older information is represented in a much weaker form only in the norm minimisation.

Examining the computational cost, it is clear that *Restart Time* and *Build Wtil* time scale well with number of columns η in the current time groups $\boldsymbol{V}_k$ and $\widetilde{\boldsymbol{W}}_k$. The computational time does not change significantly between the QR2 and the QR3 filters, nor are they as sensitive to the number of time steps reused per group or between restarts. As the number of time steps per group increases, the computational cost of the filtering dominates over the *Restart Time* and *Build Wtil* for the QR2 filter. However, the new QR3 filter and the pre-scaling weight monitoring can drive down the cost of filtering larger $\boldsymbol{V}_k$. The filtering time with the QR2 filter is not as sensitive for the IQN-IMVJ method as compared to the IQN-ILS, as when the maximum number of time steps is reached, a new empty group is created, whereas the IQN-ILS method only removes the oldest time step. Therefore, $\boldsymbol{V}_k$ always remains at maximum size in the IQN-ILS method once the time step threshold ζ is reached, whereas for the IQN-IMVJ method, it is cleared and we start off filtering empty matrices.

3.4 Selection of Default Values

The new enhancements to the quasi-Newton methods have proven their efficacy in Section 3.3. In order to really benefit from these enhancements, "good" default values need to be provided to make preCICE easier to configure. In the following, we make recommendations based on the observations from the numerical experiments presented above.

Firstly, the pre-scaling weight monitoring method should always be selected as the default. There is no observation that this, in general, results in an increase in the number of iterations per time step (Table 3.2). However, there are clear runtime advantages that can be gained from doing so as the weights are not updated frequently, even in the worst test case from Table 3.3.

Secondly, the QR3 filter should be the filter of choice. The number of iterations per time step are similar to that of the QR2 filter, and in some cases better, but never significantly more. The computational runtime savings of the QR-decomposition and filtering time are enormous. As the method limits the number of expensive computational operations and is not merely a newer and more parallel efficient algorithm, the computational savings carry over when performing parallel simulations.

Selecting a good filter limit is always difficult, as this is highly dependent on the problem. However, our findings are similar to that of Uekermann [Uek16] for the QR2 filter, where a value of between $\epsilon_{fi\ell ter} = 0.001$ and $\epsilon_{fi\ell ter} = 0.1$ is suitable for most applications. However, $\epsilon_{fi\ell ter} = 0.001$ is too weak for some applications, and divergence cannot be avoided, and $\epsilon_{fi\ell ter} = 0.1$ is too strong, deleting too many columns. Therefore, a value of $\epsilon_{fi\ell ter} = 0.01$ is recommended as the default for unknown problems.

Finally, if using the pre-scaling weight monitoring and the QR3 filter with the IQN-IMVJ method, grouping more than one time step between updates of $\boldsymbol{J}_{prev}^{-1}$ is recommended. Regardless of the test case, using $\zeta = 2$ to $\zeta = 5$ time steps always reduced the number of coupling iterations, except for a single test case with a specific configuration of input parameters. The RS-SVD method outperforms the RS-LS in most cases when only a limited number of time steps are executed between restarts. However, by using $\zeta_{restart} = 40$ time steps between restarts, the RS-LS was able to achieve similar results to the RS-SVD method. By using the RS-LS method, the problem of finding a suitable SVD truncation threshold ϵ_{svd} and finding a time step to freeze the pre-scaling weights is avoided. Therefore, the RS-LS is a suitable default with $\zeta_{restart} = 40$, and more complex input configurations, after some parameter tuning, can use the RS-SVD method.

3.5 Summary of Chapter 3

The goals of this chapter were to propose new enhancements to the quasi-Newton acceleration scheme in preCICE, in order to reduce the computational overhead of the algorithm and find default values that work for a variety of problems.

We started by examining the computational complexity of the various algorithms used throughout the quasi-Newton update step and determined that the filtering step is the bottleneck largely contributing to the computational runtime of the update. However, this required improvements for multiple components of the quasi-Newton update algorithm. We developed a pre-scaling weight monitoring and a new QR3 filter that, when combined, are able to periodically update the scaling weights and avoid most of the previously required full QR-decompositions. These results show an order of magnitude drop in the QR filter runtime.

Secondly, the new pre-scaling and QR3 filter step allowed for the increase of the number of time steps reused for the IQN-IMVJ method. By performing several time steps between updates of $\boldsymbol{J}_{prev}^{-1}$, more columns can be stored in the current $\widetilde{\boldsymbol{W}}_k$ and $\boldsymbol{V}_k^{\dagger}$ matrices. A reduction in the number of iterations per time step was achieved without compromising the runtime per simulation.

Finally, a set of default parameters, that can be used to start any coupled simulation, is suggested. This can allow a user to run with default values that are "good" values and allows a good starting point for input parameter optimisation if necessary.

4 Data Interpolation for Simulation Coupling

R**adial basis function (RBF) interpolation** methods have proven to be an accurate method for data interpolation using point cloud information only. However, multi-physics simulations present unique data mapping challenges that need to be addressed. Firstly, the number of points on the coupling interface may be quite large, especially in the case of volume coupled simulations. Secondly, the data mapping method has no control over the distribution of points throughout the mesh. Thirdly, information on the coupled interface needs to be interpolated and exchanged in each coupling iteration, requiring potentially thousands of RBF evaluations throughout the entire simulation. This places limitations on the computational complexity and runtime for the interpolation step. Finally, the accuracy of interpolation is highly dependent on the basis function shape parameter ξ. For some local basis functions, this shape parameter is simply the support radius. Reducing the sensitivity of the interpolation accuracy on ξ or finding good default values, is necessary to enable widespread use of RBF interpolation for multi-physics data interpolation.

In Chapter 2, the RBF interpolation method was presented, followed by the implementation in preCICE and the current limitations for RBF interpolation. This chapter explores methods to reduce the computational complexity of the RBF interpolation construction and evaluation for multi-physics simulations, and experiments are performed to find good default values. Firstly, methods to measure the RBF interpolation error are discussed. Secondly, the sources of instability and interpolation errors are presented in Section 4.3. The partition of unity method is introduced in Section 4.4, which decomposes the global domain into local sub-domains, and alleviates most of the problems of the previous section. The software implementation of the partition of unity method, as implemented in PyRBF, is presented in Section 4.5 and Section 4.5.1. Finally, numerical experiments are presented and discussed in Section 4.6.1 for surface and volume interpolation.

4.1 Radial Basis Function

As covered in Section 2.3, RBFs are built by a summation of basis functions

$$s(\boldsymbol{x}) = \sum_{j=1}^{N_{\Gamma_1}} \gamma_j \cdot \varphi\left(\| \boldsymbol{x} - \boldsymbol{x}_j^{\Gamma_1} \|_2, \xi\right) + q(\boldsymbol{x}),$$

A set of coefficients $\gamma_i \in R, i = 1, ..., N_{\Gamma_1}$ is found in order to reproduce the known solution at the input mesh, i.e.,

$$s(\boldsymbol{x}_i^{\Gamma_1}) = v_i^{\Gamma_1}, \quad \forall i = 1, ..., N_{\Gamma_1}.$$

Each vertex on the output mesh $\boldsymbol{\Gamma}_2$ can be evaluated using the RBF system to determine $s(\boldsymbol{x}_i^{\Gamma_2}), \quad \forall i = 1, \ldots, N_{\Gamma_2}$.

The work contained in this chapter aims to reduce the error of the RBF interpolation method, while simultaneously reducing the computational cost for building and solving the RBF interpolation system. The first step to achieve this is to understand how to measure the interpolation error, and which sources of error we observe. Only once this is complete can methods to reduce the error be determined, followed by reducing computational cost.

4.2 Error Estimation

In order to reduce the interpolation error for the RBF mapping algorithm, a metric for measuring the mapping error must be defined. Estimating the error using the RBF method is non-trivial even for small meshes in the order of 10^3 vertices and becomes more complex when considering large distributed meshes in the order of 10^6 vertices. Error estimation for RBFs is a key component when performing support radius (or shape parameter) optimisation, which is a frequently researched topic [Fas07]; [Sch11]; [Bia17]; [Bia16]; [Cav20]; [Ali18]; [Che12]; [Muk19]; [Kar19] , but most resources in literature only use small to moderate mesh sizes that often require inverting the interpolation matrix $\boldsymbol{A}$. This is infeasible for most practical applications and does not resolve the issue of the shape parameter optimisation for larger meshes. There is also little previous work on comparing the results of different error estimation methods, and on whether they will result in the same shape parameter value for the optimised case.

Input-Output Mesh – Consistent Interpolation:

The first and simplest method to measure the interpolation error, is to apply a known test function, $g(\boldsymbol{x})$, on both the input and the output meshes, $\boldsymbol{\Gamma}_1$ and $\boldsymbol{\Gamma}_2$. The exact solution on the output mesh is now known, and the interpolation values can simply be subtracted from this known result. First, the solution for $\boldsymbol{\gamma}$ is found knowing the input mesh values

$$g\left(\boldsymbol{x}^{\Gamma_1}\right)=s\left(\boldsymbol{x}^{\Gamma_1}\right)=\sum_{j=1}^{N_{\Gamma_1}}\gamma_j\cdot\varphi\left(\|\,\boldsymbol{x}-\boldsymbol{x}_j^{\Gamma_1}\,\|_2,\xi\right)+q\left(\boldsymbol{x}^{\Gamma_1}\right), \tag{4.1}$$

The coefficients $\gamma_j \quad \forall j=1,\ldots,N_{\Gamma_1}$ are determined such that the interpolated function $s\left(\boldsymbol{x}^{\Gamma_1}\right)$ reproduces the real function $g\left(\boldsymbol{x}^{\Gamma_1}\right)$ on mesh $\boldsymbol{\Gamma}_1$. The function is then interpolated onto mesh $\boldsymbol{\Gamma}_2$ via

$$s\left(\boldsymbol{x}^{\Gamma_2}\right)=\sum_{j=1}^{N_{\Gamma_1}}\gamma_j\cdot\varphi\left(\|\,\boldsymbol{x}^{\Gamma_2}-\boldsymbol{x}_j^{\Gamma_1}\,\|_2,\xi\right)+q\left(\boldsymbol{x}^{\Gamma_2}\right). \tag{4.2}$$

As the test function is known, the interpolated values on $\boldsymbol{\Gamma}_2$ can be subtracted from the exact function values

$$e_i^{\Gamma_2}=g\left(\boldsymbol{x}_i^{\Gamma_2}\right)-s\left(\boldsymbol{x}_i^{\Gamma_2}\right) \quad \forall i=1,\ldots,N_{\Gamma_2}, \tag{4.3}$$

where $e_i^{\Gamma_2}$ is the error at each point on the output mesh $\boldsymbol{\Gamma}_2$. The total Root Mean Squared Error (RMSE) [Cav18]; [Lin19] is used to estimate the error,

$$RMSE=\sqrt{\frac{1}{N_{\Gamma_2}}\sum_{i=1}^{N_{\Gamma_2}}\left|g\left(\boldsymbol{x}_i^{\Gamma_2}\right)-s\left(\boldsymbol{x}_i^{\Gamma_2}\right)\right|^2} \tag{4.4}$$

and the maximum absolute error (MAE), use by [Cav18],

$$MAE=max\left|g\left(\boldsymbol{x}_i^{\Gamma_2}\right)-s\left(\boldsymbol{x}_i^{\Gamma_2}\right)\right| \tag{4.5}$$

Furthermore, the relative RMSE (RRMSE) is a more appropriate measure for scenarios where the interpolation values are extremely small, and the absolute interpolation value approaches the error threshold

$$RRMSE=\sqrt{\frac{1}{N_{\Gamma_2}}\sum_{i=1}^{N_{\Gamma_2}}\frac{\left|g\left(\boldsymbol{x}_i^{\Gamma_2}\right)-s\left(\boldsymbol{x}_i^{\Gamma_2}\right)\right|^2}{\left|g\left(\boldsymbol{x}_i^{\Gamma_2}\right)\right|^2}} \tag{4.6}$$

Once the appropriate error estimation method is defined, it is possible to simply modify the shape parameter until the MAE, RMSE or RRMSE reaches a minimum value. However, this optimal shape parameter is potentially dependent on the test function used. When performing black-box interpolation without a known function on the input mesh, it is not possible to determine the error on the output mesh.

Input-Output Mesh – Conservative Interpolation:

The error estimation for consistent interpolation does not hold for conservative interpolation, as the integral of surface vectors must remain equal $\sum_{i=0}^{\Gamma_1} g(\boldsymbol{x}_i^{\Gamma_1}) = \sum_{j=0}^{\Gamma_2} f(\boldsymbol{x}_j^{\Gamma_2})$ instead of matching function values at single points. Lindner [Lin19] provided a review of error metrics for conservative error estimation and provides a comparison of weighted error, ϵ_w, rescaled interpolant error, ϵ_r, and the pointwise error from the consistent error formulation. In the context of multi-physics simulations, the weighted error ϵ_w appears the most appropriate measure, which scales the interpolated values by the ratio of the number of vertices on each mesh, defined as

$$\epsilon_{w,i}^{\Gamma_2} = s(\boldsymbol{x}_i^{\Gamma_2}) \cdot \frac{|N_{\Gamma_1}|}{|N_{\Gamma_2}|} - g(\boldsymbol{x}_i^{\Gamma_2}). \tag{4.7}$$

where $\epsilon_{w,i}^{\Gamma_2}$ is the error on the output mesh, $s(\boldsymbol{x}^{\Gamma_2})$ is the interpolation values, and $g(\boldsymbol{x}^{\Gamma_2})$ are the known test function values. Since the weight scaling is applied to the interpolated result, this may introduce a new source of error. Therefore, the consistent mapping formulation should be used to determine a suitable shape parameter.

Leave one out cross validation

An alternative approach for error estimation is the leave-one-out-cross-validation (LOOCV) method. This method requires only the input mesh $\boldsymbol{\Gamma}_1$ and the known values for the input mesh, $\boldsymbol{v}_{\Gamma_1}$. The idea behind the LOOCV method is to remove one data point $\boldsymbol{x}_i^{\Gamma_1}$ from the interpolation function, denoted as $\boldsymbol{S}^{[i]}$ using all other remaining points (i.e., $\boldsymbol{S}^{[i]}$ is the interpolant built without point $\boldsymbol{x}_i^{\Gamma_1}$ in the set $\boldsymbol{\Gamma}_1$), and determine the error of the interpolated value $\boldsymbol{S}^{[i]}\left(\boldsymbol{x}_i^{\Gamma_1}\right)$ against the known value $v_i^{\Gamma_1}$ at the point $\boldsymbol{x}_i^{\Gamma_1}$,

$$e_i^{\Gamma_1} := v_i^{\Gamma_1} - \boldsymbol{S}^{[i]}\left(\boldsymbol{x}_i^{\Gamma_1}\right), \quad i = 1, \ldots, N_{\Gamma_1}. \tag{4.8}$$

The error vector $\boldsymbol{e} = \left(e_1, e_2, \ldots, e_{N_{\Gamma_1}}\right)^T$ is formed from equation 4.8 by repeating the process for each vertex on the input mesh. This method can be very expensive, requiring $\mathcal{O}\left(N_{\Gamma_1}\right)$ solves of the RBF problem with $N_{\Gamma_1} - 1$ vertices each, which if computed using direct methods, may result in a computational complexity of $\mathcal{O}(N_{\Gamma_1}^4)$. Rippa [Rip99] proposed a simplification for the LOOCV method to solve for the error at every point in the input mesh $\boldsymbol{\Gamma}_1$ simultaneously by:

$$e_i^{\Gamma_1} = v_i^{\Gamma_1} - \boldsymbol{S}^{[i]}\left(\boldsymbol{x}_i^{\Gamma_1}\right) = \frac{\gamma_i}{\boldsymbol{A}_{ii}^{-1}} \tag{4.9}$$

where $\boldsymbol{A}_{ii}^{-1}$ is the diagonal component of the inverse of $\boldsymbol{A}$ for vertex i. This requires the computation of the inverse of $\boldsymbol{A}$, which is prohibitively expensive for any large mesh. Despite this limitation, many researchers use the LOOCV method, but usually applied to small mesh sizes ($N_{\Gamma_1} < 10^4$) [Udd14]; [Yao15]; [Cav20]. The main advantage of the LOOCV method is that the error can be determined even when the function across the input mesh is unknown, as only the pointwise data are required. However, there are two main disadvantages of the LOOCV when viewed in the context of multi-physics simulations. Firstly, if the LOOCV method is used to optimise the shape parameter, many optimisation steps could potentially be required. If the values across the input mesh interface change significantly between time steps for a time-dependent problem, multiple LOOCV optimisation procedures would be required, increasing the runtime of the coupled simulation. Secondly, this method does not account for the distribution of the output mesh $\boldsymbol{\Gamma}_2$, which is ultimately where the interpolation accuracy is to be measured. If both input and output mesh are of similar size and distribution, the LOOCV method may provide a good estimation of the interpolation quality on the output mesh. If, however, the input mesh and output mesh are non-uniformly distributed across the coupled interface, the shape parameter that minimises the error on the input mesh might not be the optimal shape parameter to minimise the error on the output mesh.

4.3 Key factors for interpolation quality and accuracy

A key performance goal for RBF interpolation is to reduce the error as far as reasonably possible. Therefore, the role of parameters influencing the error needs to be understood. The most important influencing factors for the interpolation error are:

1. shape parameter or support radius ξ,
2. the degree of non-uniformity of the interface meshes,
3. the ill-conditioning of the interpolation matrix,
4. the solver tolerance (applicable for iterative linear algebra solvers).

Shape parameter or support radius: The first factor impacting the interpolation quality is the shape parameter or support radius ξ for the basis functions introduced in Equation 2.31. Many sources in literature refer to the shape parameter of the basis function. However, in this work, we focus on compact support basis functions and Gaussian basis functions with a cut-off below a given threshold and, thus, the support radius that a basis function is non-zero for. The support radius determines the sphere of influence around each vertex in the input mesh $\boldsymbol{\Gamma}_1$ to build the interpolation matrix. Similarly, each vertex $\boldsymbol{x}_i^{\Gamma_2}$ on the output mesh $\boldsymbol{\Gamma}_2$ is influenced by all input vertices within the support radius, which we denote as ξ in the following. If using global basis functions (such as the thin-plate-splines), every vertex in the mesh is a function of all other vertices. Increasing the support radius tends to reduce the interpolation error down to a minimum before the error increases with increasing support radius. Increasing the support radius also comes at the cost of increased computational time, increased memory footprint, and eventually, ill-conditioning of the interpolation matrix.

The support radius is often chosen as a function of the mesh width h, i.e., distance between vertices in the mesh. The mesh width is simple to define for uniform grids, however this metric starts to become questionable for unstructured or adaptive meshes. The support radius can be

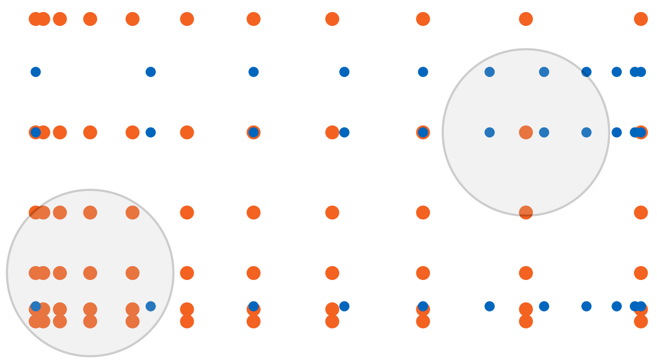


FIGURE 4.1 Support radius optimisation for compact or cut off radial basis function interpolation: example of two sets of structured vertices skewed towards opposite sides of the domain. The input mesh (orange) is refined towards the bottom-left of the domain, and the output mesh (blue) is refined towards the top-right. A globally optimal support radius leads to leaves little overlap between the support of neighbouring basis function points at the top right, where the majority of output mesh vertices lies.

defined as $\xi = g(h)$ with a simple choice being $\xi = c \cdot h$, where $c > 0$. Optimisation studies have been performed where the width of the support radius is adjusted by varying the factor c until some minimal interpolation error is achieved. The size of the input mesh is the limiting factor for any local support radius optimisation. Only small input meshes ($N_{\Gamma_1} < 10^4$) can still use the LOOCV method for a variety of test functions to find a suitable generally good support radius.

Non-uniformity: The non-uniformity of the meshes $\mathbf{\Gamma}_1$ and $\mathbf{\Gamma}_2$ can greatly influence the interpolation quality and the error. Meshes that are skewed to one side have a large difference between the maximum and the minimum distance between nearest-neighbour vertices. In this case, the optimal support radius might favour one region of the input mesh $\mathbf{\Gamma}_1$, and provide a poor interpolation quality for the opposite end of the mesh. If the distribution of points in $\mathbf{\Gamma}_2$ does not follow the distribution in $\mathbf{\Gamma}_1$ and are instead skewed toward the empty region of $\mathbf{\Gamma}_1$, determining the optimal support radius becomes more difficult.

A schematic example of the non-uniform mesh problem is shown in Figure 4.1. The input mesh (orange) is refined towards the bottom left, whereas the output mesh (blue) is refined towards the top right. As more input mesh vertices live in the bottom left space, a global support radius optimisation might select a value that favours the bottom left region (minimising the RMSE). This value may lead to a large error at the top right vertices on the output mesh. Using the LOOCV support parameter optimisation as an example, only the error estimates at the input mesh are minimised. If this is done 'in average' over all input mesh vertices, ensuring a smaller error in the bottom left region would minimise the error for most vertices in the input mesh. This would lead to the scenario described where we get a large error at many output vertices due to a too small value of the support radius. However, the input-output mesh support parameter optimisation does not suffer from the same behaviour. The input-output mesh support parameter optimisation adjusts the support radius until the error, determined by Equation 4.3, reaches a minimal value. This minimal value corresponds to the actual error on the output mesh itself and is not dependent on the refinement direction of either the input mesh or output mesh.

Ill-conditioning: A well-known problem for RBF interpolation is the ill-conditioning of the interpolation matrix $\boldsymbol{A}$ [Lin19]. If the number of interface DoF N_{Γ_1} or if the support radius increases relative to the mesh width, the accuracy of the interpolation increases, but so too does the matrix condition number, κ. The matrix conditioning is a metric for the difficulty of solving the linear system. As the condition number increases, the numerical instability for the direct solver increases and the number of iterations for the iterative solver increases. Eventually, by adding more vertices to the input mesh or increasing the support radius, the accuracy convergence saturates after which no further accuracy gains are achieved. After this point, a degradation of the accuracy and robustness can be noticed. By keeping the size of $\boldsymbol{A}$ small, or sparse by using local basis functions with small support radii, the condition number κ can be reduced, minimising the instability that influences the error.

Solver Tolerance: For larger meshes, performing a direct solve of $\boldsymbol{A}\boldsymbol{\gamma} = \boldsymbol{f}$ may become infeasible, and iterative solvers are required. This introduces a source of error through the iterative solver tolerance itself. When solving the linear system, the iterative solver terminates once a specified solver tolerance threshold $\epsilon_{iterative}$ is reached (as described in Section 2.3.3). Due to this additional tolerance, the ability to accurately capture the input field is limited by the tolerance that the linear solver can achieve. The threshold $\epsilon_{iterative}$ can be set very low for small meshes. However, such a low threshold becomes more difficult to achieve as the number of vertices on the mesh increases, or as the condition number grows. This counteracts the theoretically higher accuracy provided by fine meshes and/or large support radius. Therefore, iterative solvers can benefit from reduced mesh sizes, but in that case direct solvers are preferred.

Summary: Amongst all accuracy and efficiency issues discussed, some cannot be easily remedied. The mesh non-uniformity is fixed as the vertices are supplied by the solvers. The non-uniformity may be due to each individual physics solver needing to accurately capture specific physical behaviour in a specific region. Based on the observation, that the support radius, ill-conditioning, and solver tolerance related errors can all be improved by limiting the size of the input mesh N_{Γ_1}, we propose a partition of unity method in the following section, which is a popular method for decomposing a problem into smaller sub-problems, while maintaining a global solution.

4.4 Partition of Unity

Many problems when using the RBF interpolation are due to the input mesh size: *(i)* direct solvers suffer from high computational complexity of $\mathcal{O}(N_{\Gamma_1}^3)$ in time and $\mathcal{O}(N_{\Gamma_1}^2)$ in memory, *(ii)* iterative solvers require additional solver tolerances that increases the interpolation error, *(iii)* large meshes with wide support radii cause ill-conditioning of the interpolation matrix $\boldsymbol{A}$, and *(iv)* expensive computation of $\boldsymbol{A}$ push support radius optimisation studies out of reach. Therefore, it seems only natural that decreasing the problem size helps to solve each issue mentioned in Section 4.3.

The partition of unity (PoU) approach is a popular technique in mathematics to reduce a problem to smaller, local problems that can be combined to form a global solution. These types of methods are also known as domain decomposition techniques, and have been applied to RBF interpolation [Cav16]; [Cav18]; [Cav20] and for meshless RBF solvers for PDE's [Gar18b]; [Li16]; [Mil20]; [Sha17]. Cavoretto et. al. [Cav18] determined the optimal support radius value for each local PoU problem, but this is only tested for up to 66,049 points on the global input mesh. This size is beyond the capabilities of a regular direct solver for a dense matrix $\boldsymbol{A}$, but feasible for many local problems with the PoU method or with iterative solvers. Furthermore, methods to improve the efficiency of generating the sub-domains were presented by [Cav16]. Allowing for a single, large, global mesh to be decomposed into many overlapping local domains offers numerous advantages:

1. Direct solvers can be used for each local domain (solving problem *(i)*),
2. ill-conditioning is avoided (problem *(iii)*),
3. basis functions with large support radius or even global basis functions can be used (problem *(iv)*),
4. if support radius optimisation is performed, the computational expense if far lower for each local domain (problem *(iv)*),
5. solving many local, distributed problems improves parallel scaling.

The PoU method begins by decomposing $\boldsymbol{\Gamma}_1$ into d subsets $\boldsymbol{\Gamma}_{1,j} \quad \forall j = 1, \ldots, d$, such that $\boldsymbol{\Gamma}_1 \subseteq \cup_{j=1}^{d} \boldsymbol{\Gamma}_{1,j}$. The vertices in each subset $\boldsymbol{\Gamma}_{1,j}$ are defined as $\boldsymbol{x}_{j,i}^{\Gamma_1}$ for $N_{\Gamma_1,j}$ vertices on the input mesh and $\boldsymbol{x}_{j,i}^{\Gamma_2}$ for $N_{\Gamma_2,j}$ vertices on the output mesh. The interpolation functions from these partitions are combined through compactly supported, non-negative continuous weighting functions $W_j : \mathbb{R} \rightarrow \mathbb{R}$ with

$$\sum_{j=1}^{d} W_j(\boldsymbol{x}_i) = 1 \quad \forall \quad \boldsymbol{x}_i \in \boldsymbol{\Gamma}_1 \tag{4.10}$$

Theoretically, any weighting function can be used. Cavoretto et. al. [Cav18] used Shepards weights defined as

$$W_j(\boldsymbol{x}_i) = \frac{\overline{W}_j(\boldsymbol{x}_i)}{\sum_{k=1}^{d} \overline{W}_k(\boldsymbol{x}_i)}, \quad j = 1, ..., d, \quad i = 1, ... N_{\Gamma_1}, \tag{4.11}$$

where $\overline{W}_j(\boldsymbol{x}_i)$ are compactly supported functions. As a specific choice for $\overline{W}_j$, Wendland functions are a popular choice as they are smooth, local, positive definite functions [Cav18]. The Wendland C^2 function is defined as

$$\overline{W}(\boldsymbol{x}) = (1-r)^4 (4r+1),$$

where r is the radius of the output vertex and is described in Section 4.4.1 below. Therefore, if a vertex $\boldsymbol{x}_i$ lies in multiple sub-domains, then the sum of weights for this vertex from all sub-domains equals one and satisfies Equation 4.11. Once the weights for each output vertex have been determined, the interpolated value $\overline{S}(\boldsymbol{x})$ at a vertex $\boldsymbol{x}_i^{\Gamma_2}$ is determined as a combination of the interpolated value from each partition j, multiplied by the weighting value $W_j\left(\boldsymbol{x}_i^{\Gamma_2}\right)$:

$$\overline{S}\left(\boldsymbol{x}_i^{\Gamma_2}\right) = \sum_{j=1}^{d} s_j\left(\boldsymbol{x}_i^{\Gamma_2}\right) W_j\left(\boldsymbol{x}_i^{\Gamma_2}\right), \quad \forall \boldsymbol{x}_i^{\Gamma_2} \in_2 . \tag{4.12}$$

A schematic of the overlapping subdomains with input mesh vertices (orange) and output mesh vertices (blue) is shown in Figure 4.2. The method of determining the weighting functions is dependent of the shape of the sub-domain, i.e., are the sub-domains hyperspheres or hyperrectangles. In this work, we utilise hyperspheres with a uniform radius for each hypersphere. In the following section, the procedure used to determine the output vertex weights is described, followed by the method used to decompose the input mesh and output mesh into multiple sub-domains.

4.4.1 Weighting Functions

The traditional partition of Unity approach (PoU) uses several overlapping hyperspheres, shown in Figure 4.2. All input mesh vertices that lie within one hypersphere, $\boldsymbol{\Omega}_j$, form an RBF interpolation for that hypersphere. Values at output vertices are calculated as the weighted sum of interpolation function values from all sub-domains they are contained in according to Equation 4.11.

Consider the output mesh vertex (blue vertex) in Figure 4.2 that lies in both Ω_1 and Ω_2, with weights

$$\overline{W}_1(\boldsymbol{x}^{\Gamma_2}) = \left(1 - \frac{r_1}{R_1}\right)^4 \left(4\frac{r_1}{R_1} + 1\right) \quad \text{and} \quad \overline{W}_2(\boldsymbol{x}^{\Gamma_2}) = \left(1 - \frac{r_2}{R_2}\right)^4 \left(4\frac{r_2}{R_2} + 1\right).$$

The final weighting function is then determined using the Shepards weights in Equation 4.11,

$$W_1 = \frac{\overline{W}_1}{\overline{W}_1 + \overline{W}_2} \quad \text{and} \quad W_2 = \frac{\overline{W}_2}{\overline{W}_1 + \overline{W}_2}.$$

As the output vertex approaches the boundary of a sub-domain, the corresponding weighting $\overline{W}_i$ tends to zero. This is important as the interpolation error increases near the boundary of the RBF domain (Figure 4.2 – left). The error can significantly reduce the interpolation accuracy as a vertex approaches any boundary for up to 4 sub-domains or 8 sub-domains in 2D and 3D respectively.

In this work, we introduce an additional buffer zone, which in essence acts as an extended weighting function formula that reaches zero before the sub-domain boundary. The weighting functions are determined from the over-lapping hyperspheres using the radii $R_j, \quad \forall j = 1, \ldots, d$, ensuring that the entire domain is covered. However, the RBF interpolation field is built using all input mesh vertices lying inside of a radius $\overline{R}_i, \quad \forall i = 1, \ldots, d$. Therefore, any point that lies on the boundary R_j remains far away from the sub-domain interpolation boundary of $\overline{R}_j$, and therefore still has a satisfactory error. The error at the domain boundary at R_j is shown in Figure 4.2 – left – and the new extended radius $\overline{R}_j$ – right. The black nodes in the error plot above the hyperspheres on Figure 4.2 – right – illustrate a low error value obtained at the boundary of R_j.

4.4.2 Conservative Mapping

In order to provide conservative interpolation for PoU interpolation, a global scaling function must be used as regular conservative mapping applied within each sub-domain, no longer hold for the global interpolation. In conservative mapping, the sum of values on the input mesh must equal the sum of values on the output mesh. However, as input mesh vertices may be present in multiple, overlapping sub-domains, their contribution to the global output mesh will be summed multiple times. Therefore, a conservative mapping for the PoU RBF method consists of an initial consistent mapping in each sub-domain, followed by a rescaling of all values on the global output mesh, such that

$$\sum_{j}^{N_{\Gamma_2}} s(\boldsymbol{x}_j) = \sum_{i}^{N_{\Gamma_1}} v_i^{\Gamma_1}.$$

4.5 Implementation in PyRBF

The implementation of the PoU based RBF interpolation was implemented in an open-source solver called PyRBF[1], initially created by Lindner [Lin19][2]. PyRBF is a stand-alone, python based RBF interpolation solver that can accept input and output meshes in VTK format. It decomposes the

[1]`https://github.com/KyleDavisSA/PyRBF`
[2]`https://github.com/floli/PyRBF`

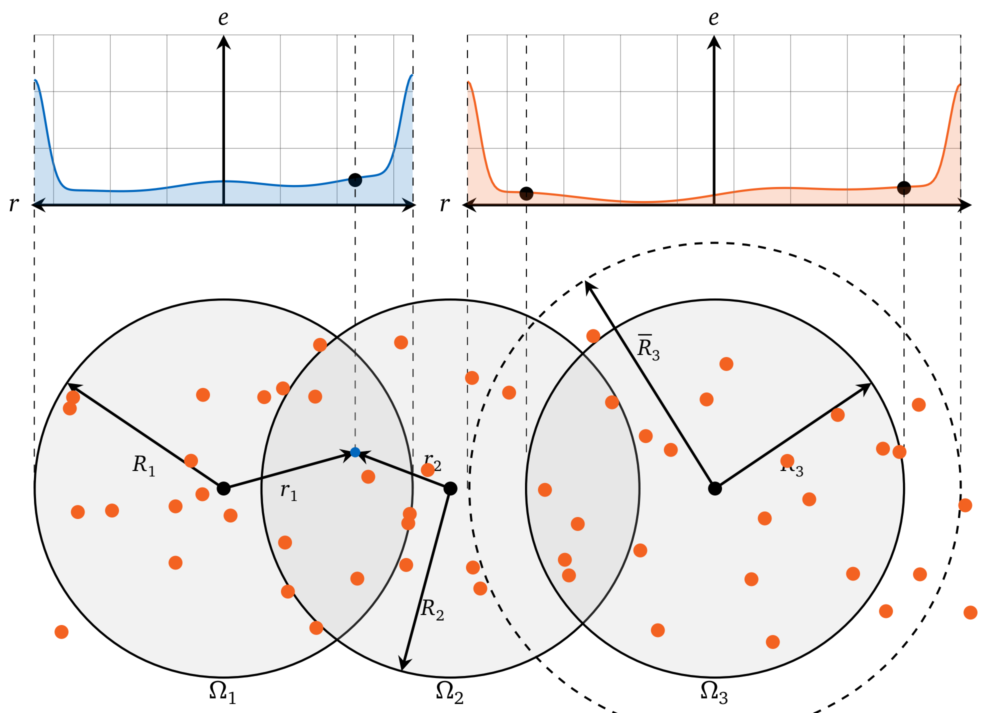


FIGURE 4.2 Partition of unity approach for radial basis function interpolation: overlapping sub-domains of the partition of unity method, illustrating the interpolation error of the interpolating functions in the respective sub-domains. The interpolation value of the output vertex (blue) is determined using the interpolation functions on both sub-domains Ω_1 and Ω_2, which are combined by the summation in Equation 4.12. The weights are determined by the ratios of $\frac{r_1}{R_1}$ and $\frac{r2}{R2}$ according to the Shepards weights in Equation 4.11. The additional buffer zone of the PoU sub-domain is illustrated by the dotted circle of radius $\overline{R}_3$ in Ω_3.

meshes into the necessary sub-domains that can be computed on distributed computing systems. The direct linear system solver utilises a QR-decomposition performed by the Numpy library[3].

PyRBF was created to provide a platform to quickly develop and test new RBF interpolation methods without being restricted to current functionality in preCICE. However, the methods developed in this thesis within PyRBF are intended to be implemented in preCICE in further projects. Selecting Python as the language of choice enabled fast development times and provided access to a vast library of mathematical functions.

In the following subsections, the parallel implementation of PyRBF is presented. The interpolation procedure is split into two core functions: domain decomposition and domain interpolation. The domain decomposition performs the PoU decomposition and assigns input mesh and output mesh vertices to the individual sub-domains, whereas the domain interpolation only performs the RBF interpolation.

[3]`https://numpy.org/doc/stable/reference/generated/numpy.linalg.qr.html`

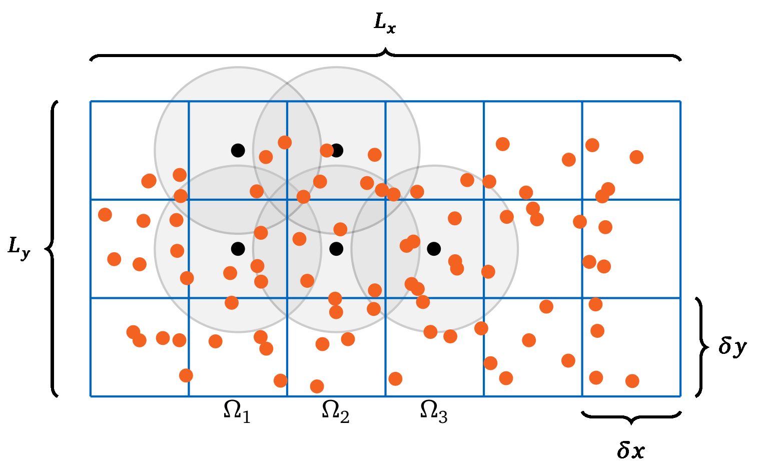


FIGURE 4.3 Partition of unity method for radial basis function interpolation: domain decomposition of a set of vertices in 2D. The bounding box (L_x, L_y) is divided into boxes of sidelength δx and δy. The PoU centres are located at the centre of each block, with the radius R defining the respective sub-domain extending into all surrounding boxes to form a complete covering of the set of vertices.

4.5.1 Parallelisation

For the PoU method to offer real computational runtime improvements, the interpolation step must be highly parallelised. It should not only scale well with N_{Γ_1} and N_{Γ_2}, but also with the number of computing ranks N_{ranks}. Due to the stand-alone functioning of PyRBF, the PoU procedure could be divided into two separate solver calls: *(i)* domain decomposition and *(ii)* RBF interpolation. Note that throughout the explanation, we refer to the three dimensions in space as the x,y and z dimension (instead of the x_1, x_2 and x_3 notation in Section 2.3.2). The method, however, generalises to any dimension.

4.5.2 Domain Decomposition

The domain decomposition begins by reading in VTK mesh files for the global input and output meshes, consisting of point cloud vertex information. The global mesh is stored in memory for each rank that the decomposition is performed on. Before the mesh can be decomposed, the spherical sub-domains and their centre points need to be defined. The sub-domain centre points are determined by measuring the bounding box of the global mesh and dividing each bounding box length L_x, L_y and L_z by the number of respective sub-domain division Δx, Δy and Δz (user defined, but automated methods based on the number of vertices per sub-domain can be implemented). The PoU sub-domain centres, $\boldsymbol{pCentre}_j = [x_j, y_j, z_j], \quad \forall j = 1, \ldots, d$, are located at the centre of each block of this decomposition, depicted by the black centre nodes in Figure 4.3.

To easily access the input and output mesh data, a `kd-tree` is built for each such that the

```
Input:  input mesh VTK: $\mathbf{\Gamma}_1$, output mesh VTK: $\mathbf{\Gamma}_2$, $\Delta x$, $\Delta y$, $\Delta z$, $c_{overlap}$
Output:  $\mathbf{\Gamma}_{1,j}$, $\mathbf{\Gamma}_{2,j}$ (input and output mesh per sub-domain)
Load mesh $\mathbf{\Gamma}_1$ and $\mathbf{\Gamma}_2$ onto each rank
Create a kd-Tree for both meshes
Determine bounding box:  $L_x$, $L_y$, $L_z$
Divide the region into separate blocks of size $\delta x = \frac{L_x}{\Delta x}$ , $\delta y = \frac{L_y}{\Delta y}$ , $\delta z = \frac{L_z}{\Delta z}$
Create PoU centre points $\boldsymbol{pCentre}_j$ at each block and set radii $R$ and $\overline{R}$ for
each PoU
for $j = 1, \ldots, N_{d,rank}$ do
  Find all vertices $\mathbf{\Gamma}_{1,j}$ within $\overline{R}$
  Find all vertices $\mathbf{\Gamma}_{2,j}$ within $R$
  Store the mesh for each sub-domain according to global ordering of $j$
```

ALGORITHM 4.1 Pseudo-code for the RBF-PoU domain decomposition method.

mesh information is easily accessible. A kd-tree is built for both $\mathbf{\Gamma}_1$ and $\mathbf{\Gamma}_2$ by using SciPy [Vir20][4]. After creating the *kd-Tree*, the function `KDTree.queryPt` returns all points within a specified radius of another point. The computational complexity of a *kd-Tree* can be of the order:

1. $\mathcal{O}(k \cdot n \cdot \ell og(n))$ to build a full kd-tree,
2. $\mathcal{O}(\ell og(n))$ insert a new point into a balanced kd-tree,
3. $\mathcal{O}(\ell og(n))$ to find a single nearest neighbor vertex.

Once the centre points are known, each rank is assigned the sub-domains to compute the local decomposition for. Each rank is assigned $N_{d,rank} = \lceil \frac{d}{N_{ranks}} \rceil$ sub-domains, where d is the global number of sub-domains, apart from the last rank which is assigned $d - (N_{ranks} - 1)\lceil \cdot \frac{d}{N_{ranks}} \rceil$ sub-domains. The sub-domains are numbered first in the x dimension, followed by the y and then the z dimension, to generate a globally unique list of sub-domain IDs. This enables each rank to independently determine which sub-domains it must build the local mesh for. The radius $\overline{R}$ is scaled according to a user defined overlap value, where $\overline{R} = c_{overlap}R$. By default, R is $1.5 \cdot \max\left[\frac{\delta x}{2}, \frac{\delta y}{2}, \frac{\delta z}{2}\right]$, such that the entire block is covered.

Each rank uses the known centres $\boldsymbol{pCentre}_j$ as input to `KDTree.queryPt`, which returns all input mesh vertices $\boldsymbol{x}_i^{\Gamma_1}, i \in \mathbf{\Gamma}_{1,j}$ within radius $\overline{R}$, and output mesh vertices $\boldsymbol{x}_i^{\Gamma_2}, i \in \mathbf{\Gamma}_{2,j}$ within radius R for sub-domain Ω_j. The returned vertices are annotated with their global ID required to reconstruct the global solution in the interpolation step in addition to local IDs. The local meshes and IDs are saved to files named according to the global ordering of the sub-domains.

The bottleneck of the domain decomposition procedure are the multiple kd-trees that are built and mirrored on each rank. However, this is only required once to perform the initial decomposition, and the complete global mesh does not need to be read into memory again during the interpolation. The domain decomposition pseudo-code is shown in Algorithm 4.1.

[4]https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html#scipy.spatial.KDTree

4.5.3 Domain Interpolation

The domain decomposition procedure stores the local input and output mesh for each sub-domain separately in files. Therefore, the interpolation can be performed independently of the decomposition, with either one rank performing the interpolation for all PoU sub-domains, or they can be equally distributed with 1 sub-domain per rank. This is possible as each sub-domain is written separately to storage.

```
for $j = 1,\dots,N_{d,rank}$ do
    Input: local input mesh VTK $\mathbf{\Gamma}_{1,j}$, output mesh VTK $\mathbf{\Gamma}_{2,j}$; $L_x$,$L_y$,$L_z$,$\Delta x$,$\Delta y$ and $\Delta z$ for each rank.
    Create kd-Tree for $\mathbf{\Gamma}_{1,j}$ and $\mathbf{\Gamma}_{2,j}$
Divide the region into separate blocks of size $\frac{L_x}{\Delta x}$ , $\frac{L_y}{\Delta y}$ , $\frac{L_z}{\Delta z}$
Create centre points at each block and set radii $R$ and $\overline{R}$ for each PoU centre
for $j = 1,\dots,N_{d,rank}$ do
    Determine weights $\boldsymbol{W}_j(\boldsymbol{x}_i^{\Gamma_{2,j}})$ for each vertex
    Build $\boldsymbol{A}^j$, $\boldsymbol{C}^j$ and input values $\boldsymbol{v}^{\Gamma_{1,j}}$
    Perform a QR-decomposition $\boldsymbol{A}^j = \boldsymbol{Q}^j\boldsymbol{R}^j$
    Compute $\boldsymbol{\gamma}^j$ and rescale with $\boldsymbol{W}_j(\boldsymbol{x}_i^{\Gamma_2})$
    Create $\boldsymbol{v}^{\Gamma_2}$ and add rescaled values in global ordering scheme
    MPI.Reduce(MPI.SUM)
if main rank then
    Determine error: $\boldsymbol{e} = \boldsymbol{v}^{\Gamma_2} - \boldsymbol{f}^{\Gamma_2}$
```

ALGORITHM 4.2 Pseudo-code for the RBF-PoU interpolation method.

The interpolation begins with each rank loading the local sub-domain meshes into memory. The master rank also loads the complete output mesh for determining the global error. The weighting function value for each output mesh vertex is determined according to Equation 4.11. For each computing rank, the interpolation matrices $\boldsymbol{A}^j$ for the local subdomain systems

$$\boldsymbol{A}^j\boldsymbol{\gamma}^j = \boldsymbol{v}^{\Gamma_{1,j}}, \quad \forall j = 1,\dots,N_{d,ranks}$$

are created. Once $\boldsymbol{A}^j$ is built, a QR decomposition is formed for each domain forming $\boldsymbol{Q}^j\boldsymbol{R}^j$, with a computational complexity of $\mathcal{O}((N^{\Omega_{1,j}})^3)$ per sub-domain, where $N^{\Omega_{1,j}}$ is the number of input mesh vertices in sub-domain j. Therefore, even in serial, the computational complexity reduces linearly with the number of sub-domains $\mathcal{O}(d \cdot (N^{\Omega_{1,j}})^3)$. The solution $\boldsymbol{\gamma}^j$ is found by solving $\boldsymbol{R}^j\boldsymbol{\gamma}^j = -\boldsymbol{Q}^{j\,T}\boldsymbol{v}^{\Gamma_{1,j}}$ via a back-substitution step. Finally, the value at each vertex is multiplied by the weighting function value within each sub-domain, i.e.,

$$\boldsymbol{v}^{\Gamma_{2,j}} = \boldsymbol{C}^j\boldsymbol{\gamma}^j,$$

$$\boldsymbol{v}_i^{\Gamma_{2,j}} = \boldsymbol{v}_i^{\Gamma_{2,j}} \cdot \boldsymbol{W}_j(\boldsymbol{x}_i^{\Gamma_{2,j}}), \quad \forall i = 1,\dots,N^{\Gamma_{2,j}},$$

where $\boldsymbol{C}^j\boldsymbol{\gamma}^j$ is defined as in Equation 2.36. In order to prepare the generation of the global output vector by accumulating the already weighted local subdomain contributions, a zero vector $\boldsymbol{v}^{\Gamma_2} \in R^{N_{\Gamma_2}}$ is created on each rank, and the local portion of the global solution is added using the know global `ID` locations for each value in $\boldsymbol{v}_i^{\Gamma_{2,j}}$. The global solution is reconstructed by using `MPI.Reduce(MPI.SUM)`, thus summing up all values from all ranks at the main rank. The PoU interpolation pseudo-code is given in Algorithm 4.2.

To determine the error of the interpolation using the input-output method, the global output mesh on the main rank can determine the real solution of a predefined test function $\boldsymbol{f}^{\Gamma_2}$ and subtract it from the solution $\boldsymbol{v}^{\Gamma_2}$. If using the LOOCV method, the LOOCV error can be determined per sub-domain.

4.6 Numerical Evaluation of PyRBF

4.6.1 Experimental Setup

Numerical experiments of the PoU RBF interpolation method as implemented in PyRBF are performed in the following section. The main objectives are to evaluate the interpolation accuracy, computational complexity and runtime of the new PoU RBF interpolation. A second objective is to find a set of "good" RBF input configuration parameters to enhance the usability of RBF interpolation. All software and data required to reproduce the results below can be found in the data repository of the University of Stuttgart (DaRUS) dataset "Replication Data for: Radial basis function interpolation with partition of unity for PyRBF" [Dav22d].

Meshes: Firstly, the performance of the PoU RBF interpolation is evaluated on a set of unit square domain (length 1 unit in each dimension) meshes of varying resolutions. For each unit square mesh, an unstructured mesh was generated by specifying the average mesh width (edge length between vertices) varying between $h = 0.1$ and $h = 0.01$ using GMSH [Geu09]. The average mesh width, the number of vertices and test series are shown in Table 4.1, with an example unstructured mesh shown in Figure 4.4 (bottom right). The coarse test series is small enough to solve with a direct solver without the PoU method, whereas the fine series contains too many vertices per mesh to be feasible in the sequential mode. In order to demonstrate the capability of the PoU method, a volume interpolation is performed in addition to the simple 2D square domain problem. For this, a unit cube domain is used, meshed with GMSH.

Test Functions: To fully test the applicability of the PoU method for data mapping in multi-physics applications, the test functions need to range from smooth to highly oscillatory. The test functions used in this work are *(i)* a relatively smooth function f_1, *(ii)* a moderately oscillating and more realistic function f_2, as used in [Cho22], and *(iii)* a highly oscillating function f_3, each defined as:

TABLE 4.1 Numerical test case for the partition of unity method in RBF mapping: unit square meshes with varying levels of refinement. The average mesh width, the number of vertices and the test series are displayed. The coarse series can be run using a direct solver without partition of unity. The fine meshes cannot be used with a direct solver as they are too large.

h	Vertices	Series	h	Vertices	Series
0.1	153	coarse	0.005	53 240	fine
0.05	552	coarse	0.004	82 983	fine
0.04	901	coarse	0.003	148 362	fine
0.03	1 574	coarse	0.001	1 322 027	fine
0.02	3 417	coarse			
0.015	6 048	coarse			
0.013	8 007	coarse			
0.012	9 473	coarse			
0.01	13 337	coarse			

$$f_1 = 16x_1x_2(1-x_1)(1-x_2),$$

$$f_2 = 0.78 \cdot cos(10 \cdot (x_1 + x_2 + x_3)),$$

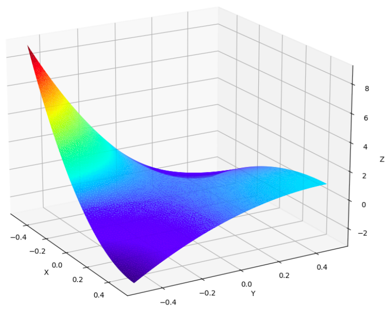
$$f_3 = 7.8 \cdot cos(15x_1) \cdot sin(15x_2) + 20.$$

Function f_1 is very smooth even relative to the mesh width of the coarsest mesh and should not present any difficulties with large support radii for local basis functions. However, function f_3 is highly oscillatory even relative to the mesh width of the medium sized meshes and may present difficulties for large support radii. For all test cases, the interpolation error is evaluated by the root mean square error (RMSE), using the input-output mesh method for consistent mapping.

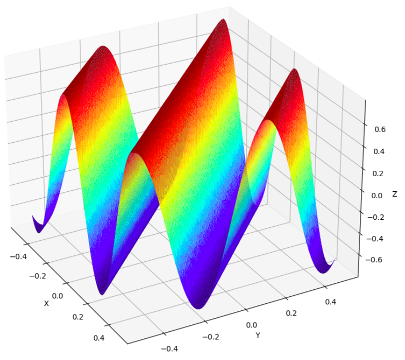
Hardware: All mapping tests were performed on the Neon computing cluster at the University of Stuttgart. The total computing resources available are: 4x Xeon E7-8880v3 (72 Core, 144 Threads), 512GB RAM, Ubuntu 20.04.

4.6.2 PyRBF Validation

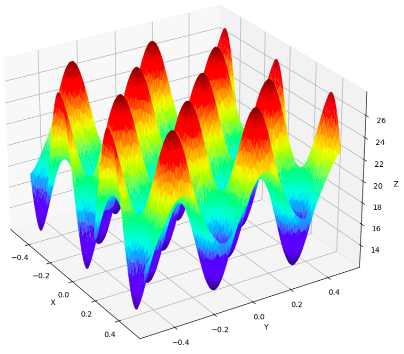
Purpose: Before testing the PoU method within PyRBF, the newly developed PyRBF solver itself must be validated. After showing that PyRBF reproduces the results from preCICE, rapid RBF interpolation development can take place in PyRBF instead. PyRBF does not offer an iterative solver and only uses a QR-decomposition in Numpy. In the following tests, PyRBF, preCICE – Eigen and preCICE – PETSc are compared using the thin-plate-splines basis function. The coarse mesh set from Table 4.1 was used as the input mesh set, where the output mesh was kept constant using $h = 0.012$ which was not included in the input mesh set.



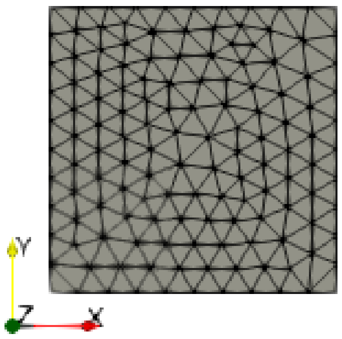
(A) F1



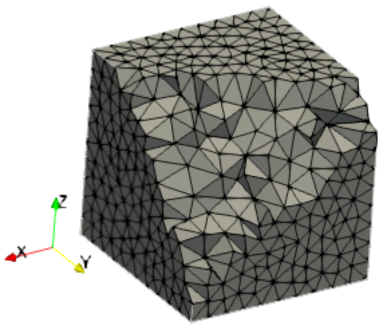
(B) F2



(C) F3



(D) Example Unstructured 2D Square Mesh



(E) Example Unstructured 3D Cube Mesh

FIGURE 4.4 Numerical test case for the partition of unity method in RBF mapping: three test functions applied to the unit square mesh. The test functions range from relatively smooth (f_1) to highly oscillatory (f_3) to cover a variety of possible scenarios encountered in multi-physics simulations. The range of test functions highlights the behaviour of varying support radii for RBF interpolation. An exemplary unstructured 2D square mesh (D) and a 3D cube mesh with a corner cut away (E) that the test functions are applied on are visualised with Paraview.

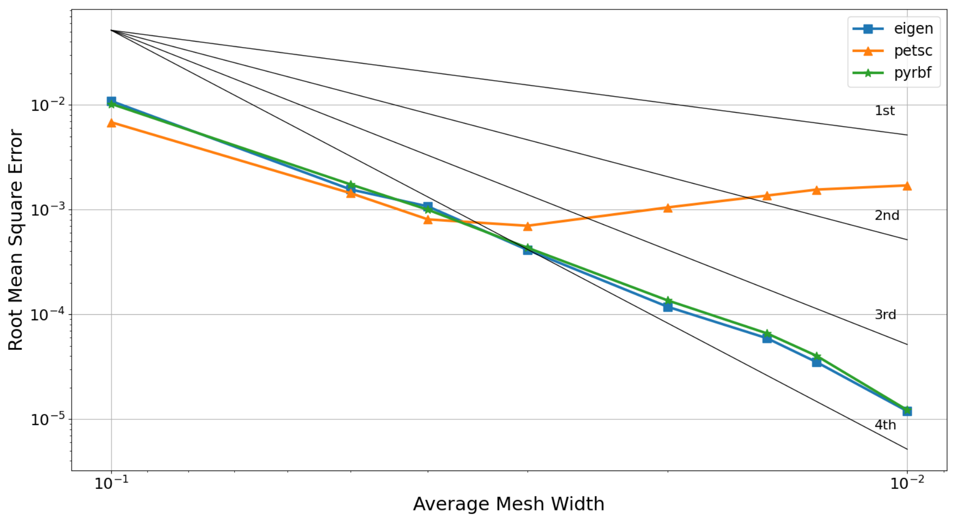


FIGURE 4.5 Numerical experiments for the PoU implementation of RBF interpolation, validation of pyRBF without PoU: global interpolation root mean square error for preCICE and PyRBF using direct and iterative solvers for the unit square mesh. The input mesh set consisted of 9 mesh with mesh width ranging from $h = 0.1$ to $h = 0.01$.

Results: A comparison of PyRBF and preCICE is shown in Figure 4.5 for varying levels of mesh refinement for the input mesh. Varying convergence orders are provided by the black lines, indication 1^{st}, 2^{nd}, 3^{rd} and 4^{th} order convergence.

Discussion: PyRBF (pyrbf) and preCICE Eigen (eigen) have almost identical results. The preCICE PETSc (petsc) error initially reduces with increasing mesh refinement, initially having a lower error than eigen and pyrbf. However, after $h = 0.04$, the error for petsc increases dramatically. The direct solvers pyrbf and eigen have an initial convergence order close to $\mathcal{O}(h^2)$, however, this improves to between $\mathcal{O}(h^3)$ and $\mathcal{O}(h^4)$ at the finest mesh resolutions.

The similarity between eigen and pyrbf is expected, as they both rely on a QR-decomposition of $\boldsymbol{A}$. Minor differences in error can be attributed to the differences in the solvers, software languages and libraries that each QR-decomposition relies on. The increase in error of the petsc solver is unexpected and it is unclear what the cause of this increase was. Various options are available to troubleshoot this behaviour within preCICE, e.g., using different preconditioners or alternative iterative solvers. However, the comparison of eigen and pyrbf is more important as these solvers are essentially equivalent. As the finest mesh had $13,337$ vertices, finer meshes could not reasonably be tested for the direct solvers. Therefore, the RBF interpolation in PyRBF is considered similar, if not the same, as in preCICE, and further development of RBF methods in PyRBF is justified.

4.6.3 Evaluation of Basis Function Choices

Purpose: Following the validation of the PyRBF solver, the behaviour of the RBF interpolation for various basis functions must be evaluated. By understanding the convergence behaviour and interpolation error of different basis functions, default values for the RBF configuration can be determined. In the following tests, the global parameterless thin-plate-splines (TPS) basis function is compared to the compact polynomial C^6 basis function from Table 2.2. The Gaussian basis function sporadically suffered from instability and was therefore excluded from the tests. The polynomial basis function support radius was varied by $r = c \cdot h$, with $c = [10, 20, 30, 40, 50]$ times the average mesh width. All tests presented in this section were performed on a single input mesh on a single compute rank. The computational cost is divided into two parts: the computational time – which is the time to perform the QR-decomposition of $\boldsymbol{A} = \boldsymbol{QR}$ – and the evaluation time – which is the time to calculate $\boldsymbol{v} = \boldsymbol{C\gamma}$.

Results: The root mean square error (RMSE), for both basis functions and all test functions, is shown in Figure 4.6. The polynomial basis function is shown as a function of the support radius, $\xi = 10h$ to $\xi = 50h$. Figure 4.7 displays the compute time (left) and evaluation time (right) for a varying number of input mesh vertices for all three test functions.

Discussion: The TPS basis function offers sufficient performance for all test functions and input mesh resolutions. However, selecting the polynomial basis with only $\xi = 20h$ already offers a significant improvement over TPS, and is already a much better alternative. The compact polynomial with $\xi = 30h$ is the best performing configuration for all test functions at the finest mesh level.

For small mesh resolutions, the difference in performance for varying ξ is less pronounced than at higher resolutions and for test functions f_2 and f_3. The error for $\xi = 40h$ and $\xi = 50h$ increases noticeably for test functions f_2 and f_3 as the mesh resolution is refined. This is most likely due to the oscillatory nature of the test functions. For test function f_1, the increase in error is only noticed at $\xi = 50h$ and for higher mesh resolutions.

The function f_1 is smoother, and therefore widening the support radius improves the solution, but also only up to a certain limit. For all test functions and mesh resolutions, optimising the support radius beyond the level performed here (increasing c in steps of 10) will most likely come at considerable computational cost for minor gains in accuracy. For the finest mesh resolution of $h = 0.01$, the support radius was 0.3 units for $\xi = 30h$. Therefore, the support of each basis function spanned more than half of the domain. For $h = 0.02$, the support already spanned the width of the domain.

Examining Figure 4.7, the timings are not very sensitive to the test function used. The compute cost grows massively for input meshes with more than $8,000$ vertices. We observe a five-fold increase from $125s$ to $537s$ for the compute time, when increasing the input mesh from $8,000$ to $13,337$ vertices. A compute time of $537s$ is too expensive to be practical for multi-

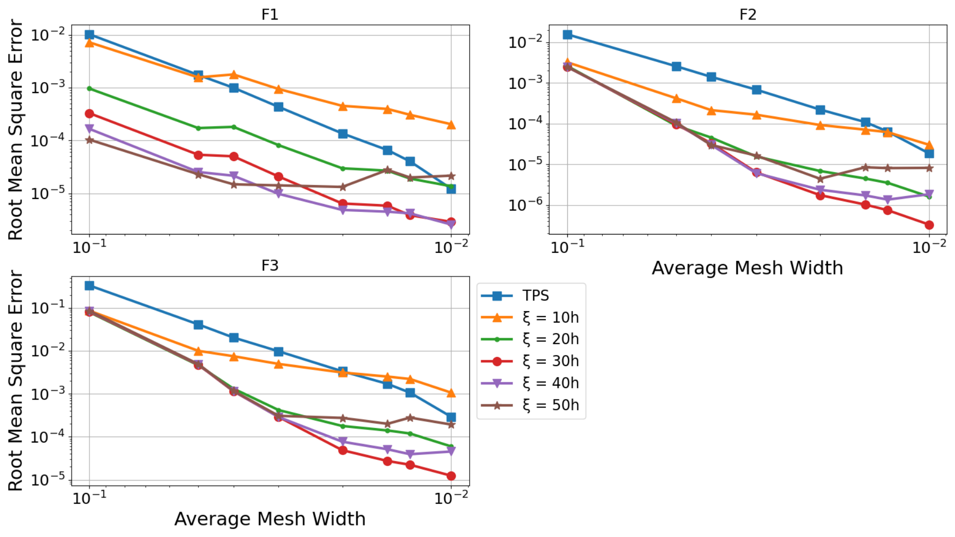


FIGURE 4.6 Numerical evaluation of different basis functions for radial basis function interpolation: root mean square error for the TPS and polynomial C^6 given with the basis function support radius ξ as multiples of the respective mesh widths h. The RMSE is plotted for each test function f_1, f_2 and f_3.

physics simulations. The evaluation time is negligible compared to the compute cost, being orders of magnitude lower than the compute cost (approximately 1,000 times lower), as expected.

4.6.4 Partition of Unity Interpolation

Purpose: In this section, the interpolation accuracy and the scalability of the partition of unity (PoU) RBF mapping are evaluated. The purpose of the PoU method is to reduce the computational cost of the mapping. Considering the runtime limit observations in Section 4.6.3, the aim is to obtain $< 6 \cdot 10^3$ DoF in each sub-domain. As the PoU sub-domain radii are defined as multiples of the maximum width in any dimension (either δx, δy or δz), having equal sized blocks for the decomposition method described in Section 4.5.2 is preferred. To analyse the PoU interpolation method, we describe the required tests in more detail.

Test *(i)*: for each test function and each basis function, the finest mesh $h = 0.01$ is divided into 1×1 (no decomposition), 2×2, 3×3, 4×4, 5×5 sub-domains ensuring equal sized sub-domains[5]. This approximately equates to an average of 13,337 (the whole mesh), 3,334 , 1,481 , 833 and 533 vertices per sub-domain, respectively. The purpose is to determine if the support radius or the number of sub-domains has a larger effect on the interpolation error.

[5]The inner sub-domains have more vertices as the boundary sub-domains extend into empty space. Therefore, equal load-balancing between sub-domains is not guaranteed.

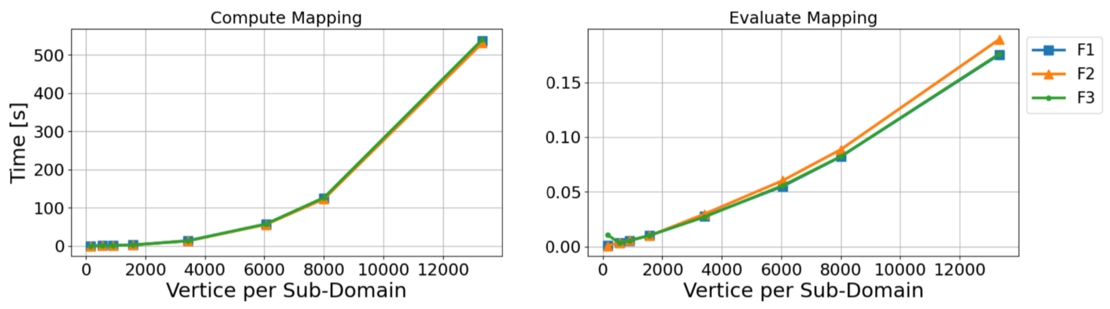


FIGURE 4.7 Numerical evaluation of different basis functions for radial basis function interpolation: compute mapping runtime and evaluation mapping runtime for the unit square mesh with mesh resolution $h = 0.1$ to $h = 0.01$. The runtime is plotted against the number of vertices for each test function f_1, f_2 and f_3.

Test *(ii)*: the sub-domain overlap is evaluated to determine the error reduction with increasing overlap. The larger overlap results in more input mesh vertices in each sub-domain, and a larger computational cost per sub-domain. However, we plot the overlap results on the same x-axis as for the no overlap case to provide a valid comparison.

Results: The RMSE for both basis functions and varying support radii ξ with varying number of vertices per sub-domain, is shown in Figure 4.8 for test functions f_1, f_2 and f_3 (for test *(i)*). The RMSE while varying the sub-domain overlap, defined as $\overline{R}/R$, is shown in Figure 4.9 (for test *(ii)*). The y-axis indicates the RMSE, and the x-axis indicates the number of input mesh vertices per sub-domain for both figures. The per-vertex errors with the f_2 test functions for the regular RBF interpolation, RBF-PoU interpolation, PoU with a finer input mesh and PoU with an extended overlap, are plotted in Figure 4.10.

Discussion: Examining Figure 4.8, the number of PoU sub-domains does not appear to influence the error nearly as significantly as the choice of basis function or support radius. The largest influence on the error is the choice of basis function or support radius, where the error fluctuates by approximately 2 orders of magnitude depending on the choice of basis function. In all test functions, the support radius $\xi = 10h$ has the largest error, followed by the TPS basis function. For test functions f_1, the support radii $\xi = 30h$ and $\xi = 40h$ have almost equal errors. However, $\xi = 30h$ has the lowest error for test functions f_2 and f_3. For the function f_2, creating more sub-domains, thereby decreasing the number of vertices per sub-domain, reduces the error for the polynomial basis functions. Reducing the size of the RBF interpolation might introduce the overlapping regions where the interpolation quality may suffer, but each RBF interpolation system is well conditioned and provides accurate results over the whole sub-domain. Increasing the support radius beyond $\xi = 40h$ results in increasing error for all test functions.

Examining Figure 4.9, the scale of the y-axis confirms that the number of sub-domains has a

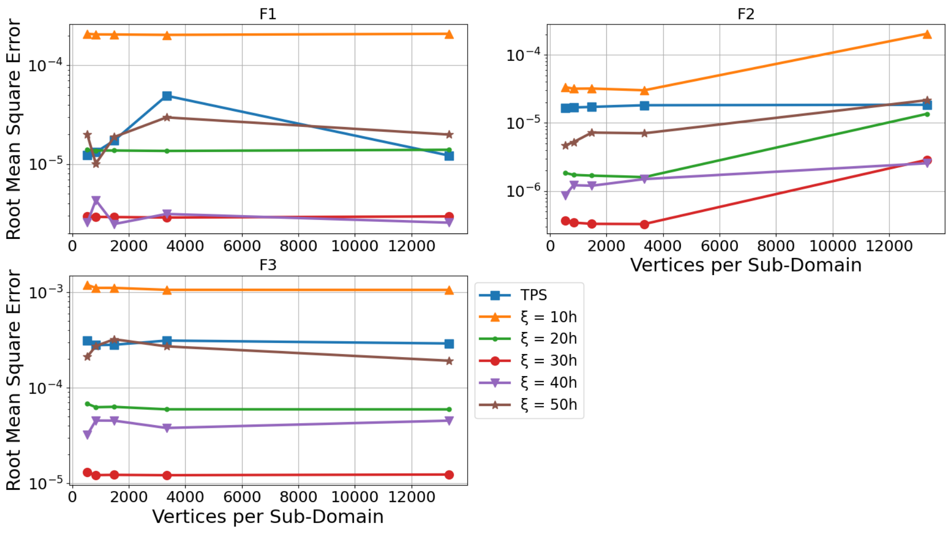


FIGURE 4.8 Numerical experiments for the PoU implementation of RBF interpolation: RMSE versus number of vertices per sub-domain, for various basis functions on the $h = 0.01$ input mesh. The RMSE is plotted for each test function f_1, f_2 and f_3.

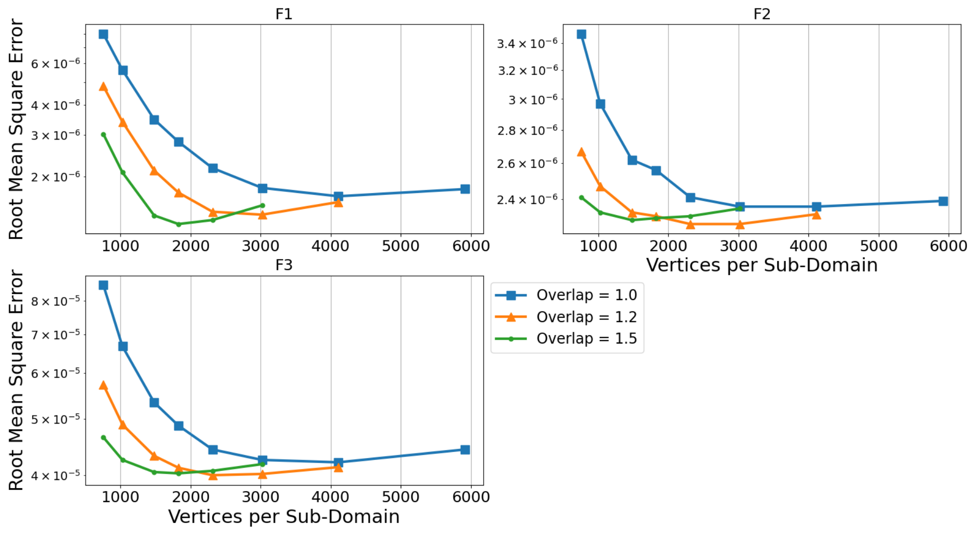
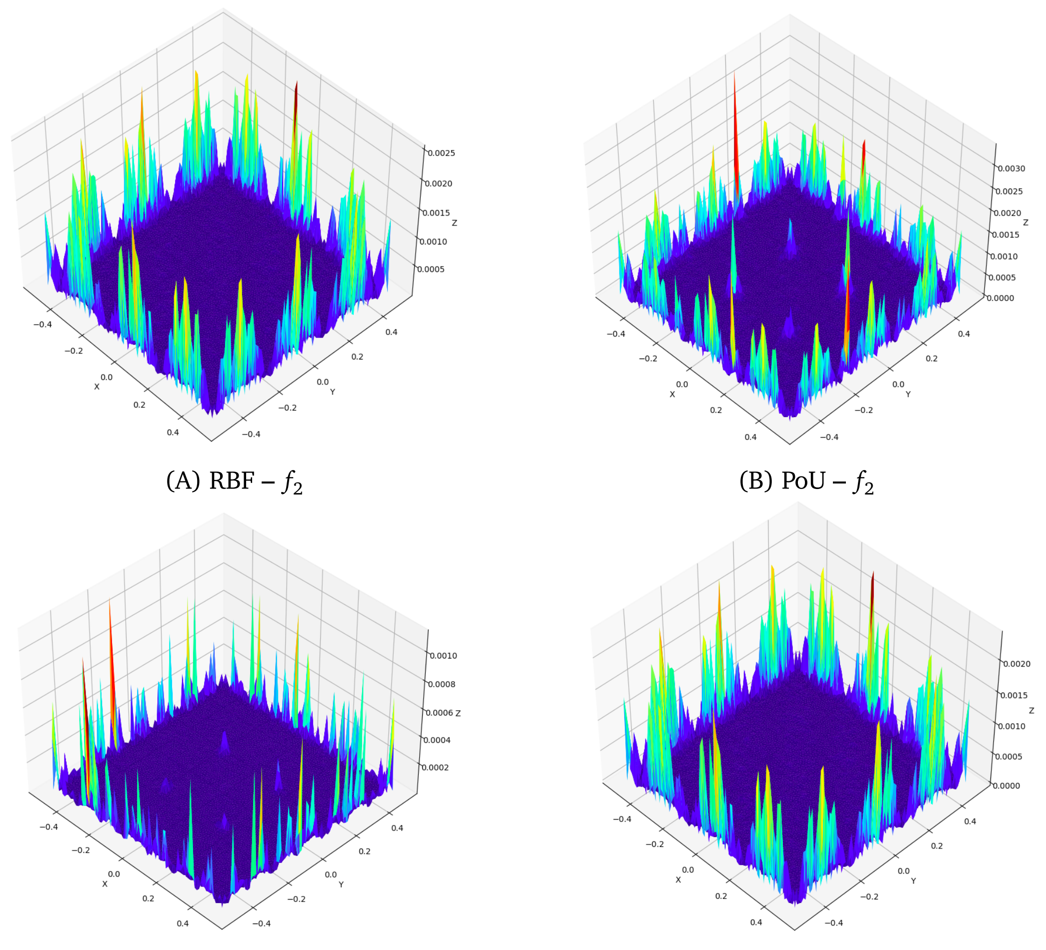


FIGURE 4.9 Numerical experiments for the PoU implementation of RBF interpolation: RMSE versus number of vertices per sub-domain, for various level of overlap. The overlap is defined as the ratio of $\overline{R}/R$. The RMSE is plotted for each test function f_1, f_2 and f_3.

much smaller impact on the interpolation accuracy than the choice of basis function or support radius. The errors vary by approximately half an order of magnitude: between 10^{-6} and $7 \cdot 10^{-6}$ for f_1, between $2.2 \cdot 10^{-6}$ and $3.4 \cdot 10^{-6}$ for f_2, and between $4 \cdot 10^{-5}$ and $9 \cdot 10^{-5}$. An overlap of 1.2 clearly out-performs the no overlap case for all test functions, but an overlap of 1.5 has a larger error than 1.2 for more than 2,500 vertices per sub-domain. However, increasing the amount of overlap comes at the expense of additional computational effort. Further extending the overlap to 1.5 appears to offer only marginal gains compared to 1.2 for a small number of vertices per sub-domain, but at considerably more computational effort. Regardless, the improvement in accuracy even for only 1,000 vertices per sub-domain, is significant and can greatly help to reduce the problem size down to a manageable level.



(A) RBF – f_2

(B) PoU – f_2

(C) PoU – f_2 Fine

(D) PoU – f_2 Overlap

FIGURE 4.10 Numerical experiments for the PoU implementation of RBF interpolation: pointwise error on the output mesh for the f_2 test functions with (A) regular RBF, (B) PoU, (C) PoU with a finer mesh and (D) PoU with overlap. All tests used the thin-plate-splines basis function and output mesh $h = 0.012$ (9473 vertices). Tests (A), (B) and (D) used the input mesh $h = 0.02$ (3417 vertices) and test (C) used the input mesh $h = 0.013$ (8007 vertices). All PoU interpolation used 9 (3×3) sub-domains.

The pointwise per vertex errors on the output mesh for the f_2 function are shown in Figure 4.10 (A) to (D). The largest errors occur at the outer boundary of the domain, consistent with the presumption in Figure 4.2. The small input mesh size of exaggerates the disadvantage of the PoU method (B), where small spikes in the output mesh error can be seen in the middle of the domain. This is caused by the error at the PoU sub-domain boundaries. The test is repeated using a finer input mesh $h = 0.013$ (8007 vertices), where a noticeable decrease in the error spikes in the middle of domain is observed but are still present. Finally, the overlap of 1.2 almost completely removes the mid-domain error spikes when observing the image. Therefore, the PoU method is suitable even for small input and output meshes when properly configured with a reasonable overlap and is necessary for large input and output meshes.

4.6.5 Computational Cost

Purpose: The PoU method is perfectly suited for parallel RBF interpolation. As the main purpose of testing new RBF methods in PyRBF is to provide a development platform, only the domain interpolation cost is evaluated here. The domain decomposition only involves the less costly `kd-tree` computation. Instead, the computational cost of the interpolation step is evaluated, and the scalability is demonstrated. We consider the computational cost while varying the number of vertices in the RBF interpolation. This can be used to give an indication of the cost per sub-domain. Test *(i)*: the first test evaluates the computational runtime for different levels of overlap while varying the number of vertices per sub-domain for test function f_1. The next three tests demonstrate the scalability of the PoU RBF method. Test *(ii – Med)*: we use the unit square mesh with $53,240$ input vertices and $82,983$ output vertices on 64 sub-domains with no overlap and use the TPS basis function. The tests were run between 1 and 36 ranks. Test *(iii – Large)*: we use the unit square mesh with $148,362$ input vertices and $1,322,027$ output vertices on 64 sub-domains with no overlap and use the TPS basis function. The tests were run between 2 and 64 ranks. Test *(iv)*: we use two volume unit cube meshes with $740,701$ input vertices and $51,204$ output vertices on 216 ($6\times6\times6$ decomposition) and 512 ($8\times8\times8$ decomposition) sub-domains with no overlap and use the TPS basis function. The tests were run from 6 to 54 ranks for 216 sub-domains, and from 4 to 64 ranks for 512 sub-domains. The decomposition into 216 sub-domains resulted in $3,429$ input vertices per sub-domain, and $1,446$ vertices per sub-domain for 512 sub-domains. A modified f_1 function was used to include the third dimension variable:

$$f_1 = 16x_1x_2x_3(1-x_1)(1-x_2)(1-x_3).$$

Results: Figure 4.11 displays the compute time (left) and evaluation time (right) for varying number of vertices per sub-domain and varying levels of overlap. The x-axis value is determined by dividing the total number of vertices on the input mesh by the number of sub-domains. This number is given for no overlap, i.e., $\bar{R} = R$. For cases with overlap, it represents only the number

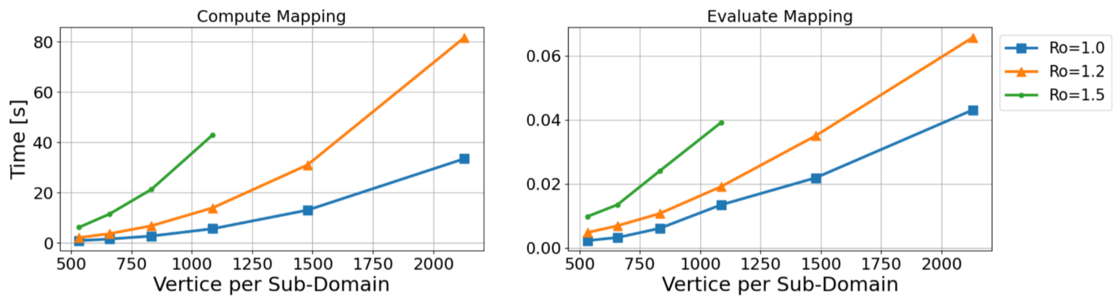


FIGURE 4.11 Numerical experiments for the PoU implementation of RBF interpolation: runtime versus number of vertices per sub-domain for the compute mapping and evaluate mapping steps, with varying overlap $R_o = \overline{R}/R$.

of vertices in the part of the sub-domain that lies with the smaller radius R.

Figure 4.12 – left – shows the strong scaling of the compute time (QR-decomposition time) and the total PyRBF solver runtime for the medium mesh $(53,240)$ and the large mesh $(148,362)$. Figure 4.12 – right – shows the scaling of the volume interpolation case for two domain decomposition sizes of 216 sub-domains and 512 sub-domains. The solid black lines indicate linear scaling.

Discussion: Examining Figure 4.11, increasing the overlap has a substantial impact on the computation time, where $\overline{R} = 1.2R$ is already twice as expensive as $\overline{R} = 1.0R$ at $1,500$ vertices per sub-domain. Overlap $\overline{R} = 1.5R$ is approximately 4 to 5 times more expensive than $\overline{R} = 1.0R$ at 1100 vertices per sub-domain.

The compute time scales almost linearly with number of ranks if the size of each sub-domain remains constant. The total PyRBF solver time (Total-Med and Total-Large) scaling immediately begins to diverge from linear scaling as the number of ranks increase. Running on only 4 ranks, the total simulation time for the large mesh with $148,362$ input vertices takes 450s to complete, whereas the regular RBF interpolation solver in Figure 4.7 with $13,337$ input mesh vertices takes 540s.

Even for volume interpolation, almost linear scaling in the number of sub-domains is apparent for the actual compute time. Once again, the total compute time deviates from the linear scaling. The number of vertices per sub-domains had almost no influence on the interpolation error, where the RMSE was 0.0100127 and 0.0102529 for 216 and 512 sub-domains, respectively.

The almost linear scaling with respect to the number of ranks was expected for the compute mapping time due to the complexity of $\mathcal{O}((\frac{1}{N_{ranks}})(\frac{N_{\Gamma_1}}{d})^3)$. However, the dependence on $N_{\Gamma_1}^3$ results in a large increase in runtime for a small increase in the extended radius $\overline{R}$. If the computation time is a severe bottleneck, not using any overlap still results in sufficiently good interpolation, but slightly increasing the overlap will improve the interpolation quality. The volume interpolation case was used to demonstrate the ability of PoU methods to reduce the computational cost for

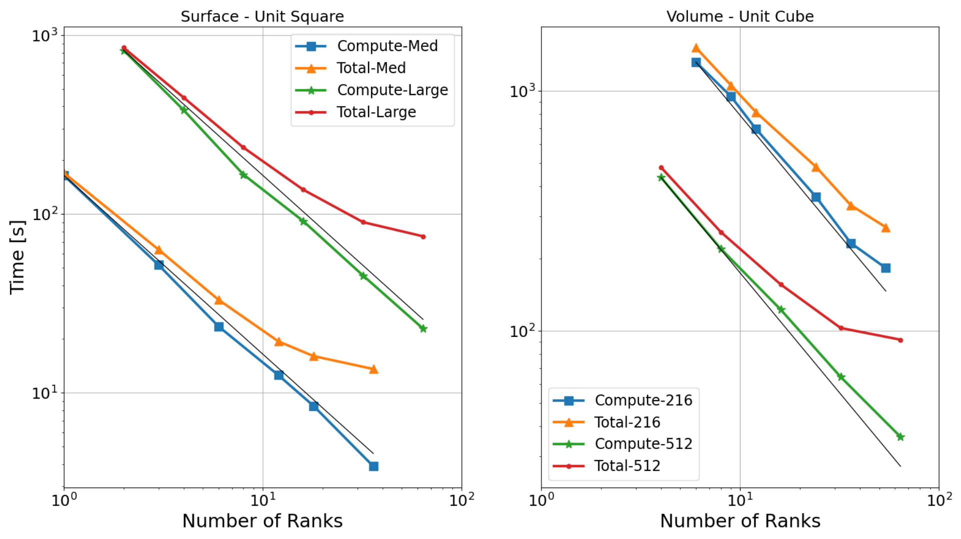


FIGURE 4.12 Numerical experiments for the PoU implementation of RBF interpolation: runtime versus number of ranks for the compute mapping and total simulation time for *(i)* surface unit square interpolation (left) and *(ii)* volume unit cube interpolation (right). The compute mapping computation scales almost linearly with the number of ranks, whereas the total PyRBF solver time is constricted by additional overhead.

not only large problems, but three-dimensional data. Including the extra dimension into the interpolation did not impact the computational time at all, and only a minor difference was notices in the interpolation error.

The total simulation time does not scale well due to constant overhead in the PyRBF solver. Firstly, the whole input and output mesh are read from file into the main rank. This is constant overhead that does not change with the number of ranks, as it is a purely serial operation. Secondly, a global `AllReduce` communication operation is performed to sum the values across all ranks into a single vector on the main rank. For implementation into a multi-physics solver, the global output data on the main rank would need to be communicated to all other ranks, or a point-to-point communication system would have to be implemented instead.

4.6.6 Default Parameter Configuration

Purpose: In the sections above, we showed that the PoU method can reduce the computational cost of the RBF interpolation while maintaining a high interpolation accuracy. In this section, we propose a default configuration for the PoU radial basis function interpolation that should provide a robust, reasonably accurate, and computationally feasible solution for a large variety of applications. Our previous observations lead us to create a so-called default basis function, where

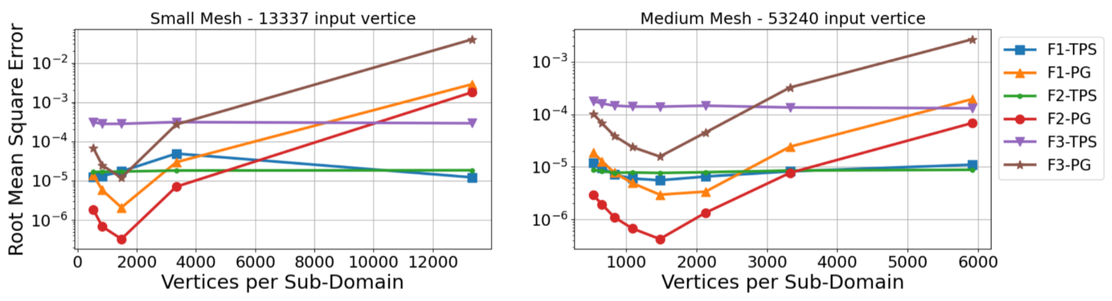


FIGURE 4.13 Numerical results for suggested default basis functions for PoU radial basis function interpolation: RMSE versus number of vertices per sub-domain for the TPS and the parameterless polynomial C^6 basis function. The basis functions were evaluated for all three test functions f_1, f_2 and f_3.

the polynomial basis function's support radius is set to the width of the PoU domain. As each sub-domain is small enough, the computational effort and memory footprint of a large support radius no longer limits the RBF interpolation. This creates a parameterless basis function for the compact polynomial C^6 basis function. We evaluate this configuration for two output meshes $h = 0.01$ and $h = 0.005$, with 13,337 and 53,240 vertices each respectively. These were evaluated using the PoU RBF method for all three test functions. No overlap was used for the tests.

Results: The RMSE versus the number of vertices per sub-domain, for both the TPS and new parameterless compact polynomial basis functions are plotted in Figure 4.13 for 13,337 output mesh vertices (left) and 53,240 output mesh vertices (right). The results are shown for all three test functions f_1, f_2 and f_3.

Discussion: For each output mesh, the minimal interpolation error occurs for the parameterless polynomial (termed polynomial global – PG) at approximately 1,500 vertices per sub-domain for each output mesh and all test functions. The TPS error is lower than the PG error for more than 5,000 vertices per sub-domain for 13,337 vertices on the input mesh, whereas the TPS error is lower than the PG error for more than 3,000 vertices per sub-domain for 53,240 vertices on the output mesh.

As the number of vertices per sub-domain of the input mesh increases substantially (greater than 5,000), the width of the support radius increases too much, resulting in an increase in the error. This is a similar pattern found in Figure 4.6. Similarly, as the number of vertices per sub-domain decreases below 1,000, the width of the support radius decreases too much, reducing the interpolation accuracy. Interestingly is the similar behaviour for both output mesh sizes and all three test functions. The optimal performance for the parameterless polynomial occurs at approximately 1,500 vertices per sub-domain, regardless of the test function or output mesh resolution.

Therefore, it can be advised that if the input mesh is decomposed into blocks with fewer than $3,000$ vertices per sub-domain, then the parameterless PG basis function is preferred, otherwise the TPS basis function offers satisfactory performance. By sticking to this default, equates to each vertex in each sub-domain is a function of potentially thousands of surrounding vertices. This should provide a good interpolation condition, where vertices far away should not influence the interpolation too much, if at all.

4.7 Summary of Chapter 4

The goal of this chapter was to evaluate issues in accuracy and computational efficiency of radial basis function interpolation, and to develop methods to reduce the error and computational costs.

We began by discussing the influencing factors for the error in the context of multi-physics simulations and identified the support radius choice of the basis functions, the non-uniformity of meshes, potential ill-conditioning of the system matrix and iterative solver tolerances as the main factors. In addition, we concluded that reducing the problem size can help alleviate all these errors.

The partition of unity method was introduced as a method to reduce the computational complexity. The global problem is decomposed into overlapping sub-domains, cheap RBF computations are performed on each sub-domain, and the local solutions are recombined to form a global solution. Varying overlap between subdomains was implemented to help improve the interpolation accuracy.

The PyRBF implementation was discussed, including the decomposition and interpolation methods. The numerical testing results showed that:

- the partition of unity reduces the problem size without significantly increasing the interpolation error,
- the partition of unity greatly reduces the computational cost of the direct solver,
- excellent scaling of the PoU combined with a direct solver was achieved,
- a good parameterless polynomial basis function was introduced and tested, with satisfactory accuracy.

Part II

Geothermal Energy Infrastructure Optimization

5 Shallow Geothermal Resource Optimisation

Addressing the challenges of the energy crisis and tackling climate change grows more important with each passing day. A large focus is on cities and urban environments, which require enormous amounts of energy to meet their growing heating and cooling needs for commercial buildings, households and industry. The 2011 Energy Efficiency Plan by the European Commission stated that buildings contribute to 40% of final energy consumption, with space heating accounting for approximately two-thirds of energy consumption in residential homes [Eur11]. In the past, space heating and cooling demands have largely been met through fossil fuel based sources. However, shallow geothermal energy has been highlighted as a viable alternative to meet the ever-growing space heating and cooling needs. **G**round**w**ater **h**eat **p**umps (GWHP) are able to transfer energy between buildings and a subsurface aquifer, providing for space heating and cooling needs. As the groundwater temperatures are relatively constant throughout the year, GWHPs heat up buildings in the cold winter and cool them down in the hot summer.

In Part II of this thesis, we address the challenges of the Geo.KW project in building a planning and optimisation tool, that can be used to predict the influence that widespread GWHP usage has on the subsurface aquifer, and to optimise the layout and usage of GWHPs on a city-wide scale. Long-term resource management of the subsurface aquifer is critical to utilise this natural resource reliably and sustainably. Current state-of-the-art models are only able to perform singular tasks when it comes to modelling the effects of GWHP usage, i.e., the subsurface is modelled according to current or predicted groundwater usage and GWHP layout plans. However, this does not allow to systematically exploit knowledge about the temperature distribution in the aquifer in the planning process of new geothermal infrastructure. Therefore, a new modelling method that combines a subsurface flow and temperature simulation with an energy infrastructure optimisation model was devised. The specific focus of this thesis tackles the development of the respective coupling environment between the subsurface simulation and infrastructure planning. This combines detailed numerical groundwater simulation and energy infrastructure optimisation together in a strongly coupled way, allowing for all software components to communicate and exchange data during the simulation and optimisation process.

In this chapter, the role that groundwater resource management has in meeting the growing heating and cooling demands in urban environments is discussed and motivated, followed by a description of the various project components. This chapter also includes the work of the GEO.KW project team that is ultimately built upon to provide the software coupling environment. Firstly, the functioning of heat pumps, and the problem of city-wide heat pump usage, are discussed in Section 5.1. Next, the planning and optimisation tool is described in Section 5.2, which defines the required outputs and objectives for developing a coupled simulation tool, followed by defining the individual software components. The numerical groundwater modelling technique, domain definition, meshing techniques and calibration, are described in Section 5.3. In Section 5.4, energy infrastructure optimisation modelling is introduced, followed by the specific optimisation approach used within this project. Methods to couple the independent simulation and optimisation solvers together, including how the parallelisation is accounted for in the coupling, is described in Chapter 6.

5.1 The Energy Beneath Our Feet

The utilisation of GWHPs to provide for space heating and cooling demands offer numerous advantages. In the following section, we describe the unique subsurface conditions within the city of Munich, and how GWHPs function to meet current and future energy demands.

5.1.1 Subsurface Heat Island

Cities and urban environments around the world suffer from a phenomenon called the heat island effect. This is where urban environments are hotter than their surrounding rural areas due to human activities and the infrastructure built onto the surface [Sol05]. Large urban structures absorb and emit the sun's heat more than natural environments such as open fields, forests and lakes. The increased temperatures above the surface can make its way down into shallow aquifers, known as the so-called subsurface urban heat island (SUHI) [Gun11]. Elevated temperatures within shallow aquifers come with numerous risks, including degradation of drinking water quality [Mül14], changes in microbial ecosystems [Gar18a] and increased contaminant transfer [Bon13].

Careful management of subsurface aquifers is necessary for cities to avoid the above mentioned risks and to not over exploit the natural resource. Studies have identified anthropogenic[1] heat sources that are largely responsible for SUHI [Men13a], especially due to the heat loss from buildings, and elevated ground temperatures from land mass use and the removal of natural spaces [Hem19]. Additionally, methods have been developed to quantify the anthropogenic heat

[1]"relating to, or resulting from the influence of human beings on nature" – `https://www.merriam-webster.com/dictionary/anthropogenic`

fluxes by analytical calculations [Men13b]. However, quantifying the human-induced temperature changes in the subsurface is difficult, especially for depths up to 10m, as they are also seasonally influenced [Ban09]. In urban areas, the seasonal variation coincides with anthropogenic heat sources, and results in a highly dynamic thermal subsurface environment [Fer07].

A unique by-product of the SUHI is that it promotes the use of shallow GWHPs to meet space heating demands, which results in a cooling of the groundwater, remediating the SUHI problem [All03]. Not only is the SUHI problem mitigated, but the dependence on fossil fuels for heating can be decreased. However, meeting cooling demands is hampered for cities, where the demand is expected to increase. Therefore, the use of GWHPs for cooling requires a precise understanding of the subsurface environment and the impact on groundwater temperatures.

5.1.2 Munich Subsurface

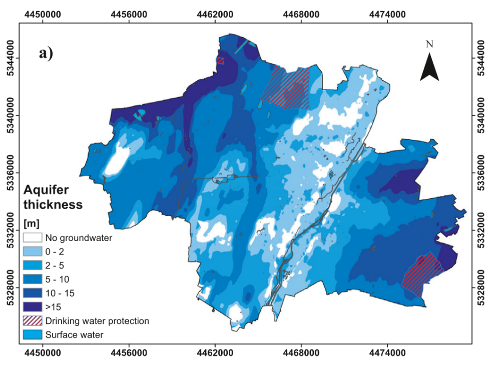
The city of Munich is the pilot study area within the GEO.KW project. The state capital of Bavaria has over 1.5 million inhabitants, a dense urban area and is one of the fastest growing cities in Germany. The city itself has not suffered substantially from the urban heat island effect, which is partially attributed to the north-south orientated river Isar which helps channel cold air into the city.

Munich lies approximately 50km north of the Alps, in the so-called Munich Gravel Plain which was deposited during the last ice age. This gravel plain is an important feature for subsurface aquifers, as it typically has very good hydraulic permeability. The groundwater in the quaternary layer flows north or north-east, depending on the slope of the terrain. Additionally, the Isar river influences the direction of the subsurface water flow. Beneath the surface, the average depth to groundwater is 7.5m. However, due to the changing height of the gravel, the groundwater level can be only 1-2m below the surface in some parts. The tertiary layer beneath is comprised of silts and clay with reduced permeability for water flow.

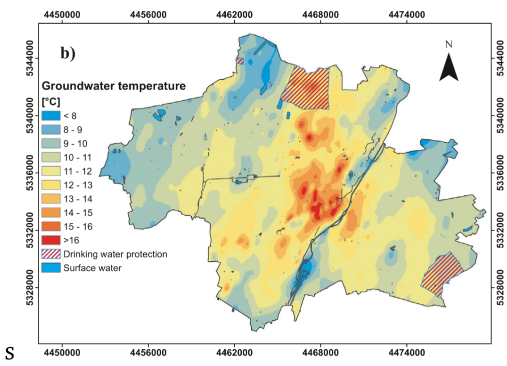
Due to the specific quaternary layer found in Munich, the city is well suited for the use of shallow GWHPs. Various groundwater conditions found in Munich are displayed in Figure 5.1. The aquifer thickness, depth-to-water and surface sealing are all factors contributing to the ease of access to the groundwater. With most of the city having a depth-to-water between 5m to 15m, access to shallow GWHP locations is available almost everywhere. The groundwater temperatures and flow velocities influence how many GWHPs can be utilised. A higher velocity results in lower peak temperatures at a GWHP injection site, however a thermal plume is generated that tends to propagate further downstream. This effect is explained in more detail below.

5.1.3 Groundwater Heat Pumps

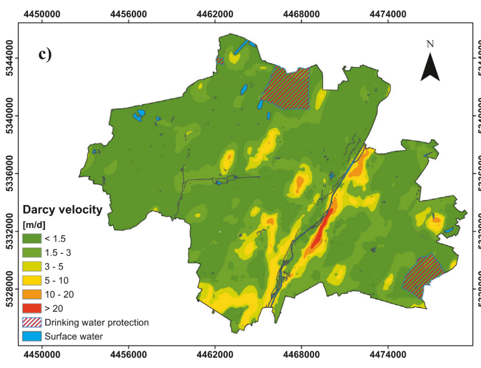
Shallow GWHPs are systems that transfer heat to and from groundwater within a shallow aquifer, typically a few meters below the surface. These GWHP systems can be divided into two main types: closed loop or open loop. Open loop systems are well suited for aquifers with relatively fast



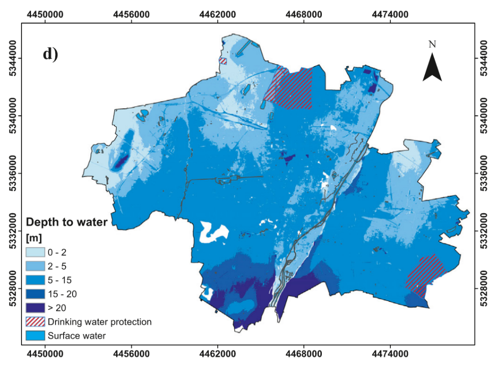
(A) Aquifer Thickness



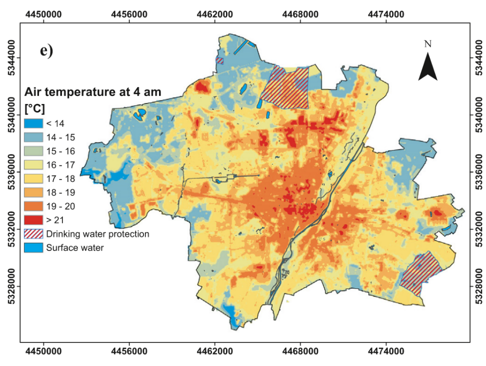
(B) Groundwater Temperature



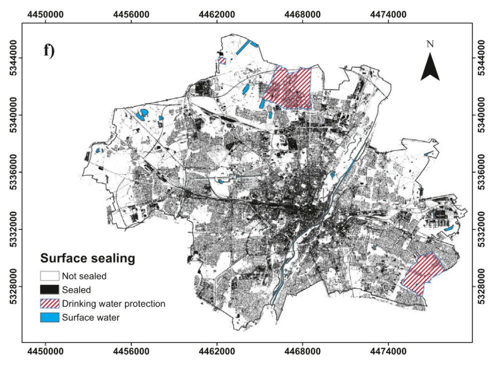
(C) Darcy Velocity



(D) Depth to Water



(E) Air Temperature



(F) Surface Sealing

FIGURE 5.1 Dataset of influential factors for shallow groundwater usage (images used with permission from Böttcher and Zosseder [Böt22]). The aquifer thickness, depth-to-water and surface sealing are important factors in determining where GWHPs can be placed. The groundwater temperatures and velocities influence the efficiency of a GWHP.

flowing groundwater and consist of three parts: *(i)* an extraction well, *(ii)* an injection well and *(iii)* a heat exchanger. A schematic of an open loop GWHP in use is shown in Figure 5.2, where the groundwater is flowing from left to right. During heating mode, i.e., when used for heating a building, relatively warm water (the water is warmer relative to the building) is extracted from the subsurface aquifer through a single extraction well or a set of extraction wells (left well in Figure 5.2), passed through a heat exchanger which extracts the heat from the groundwater,

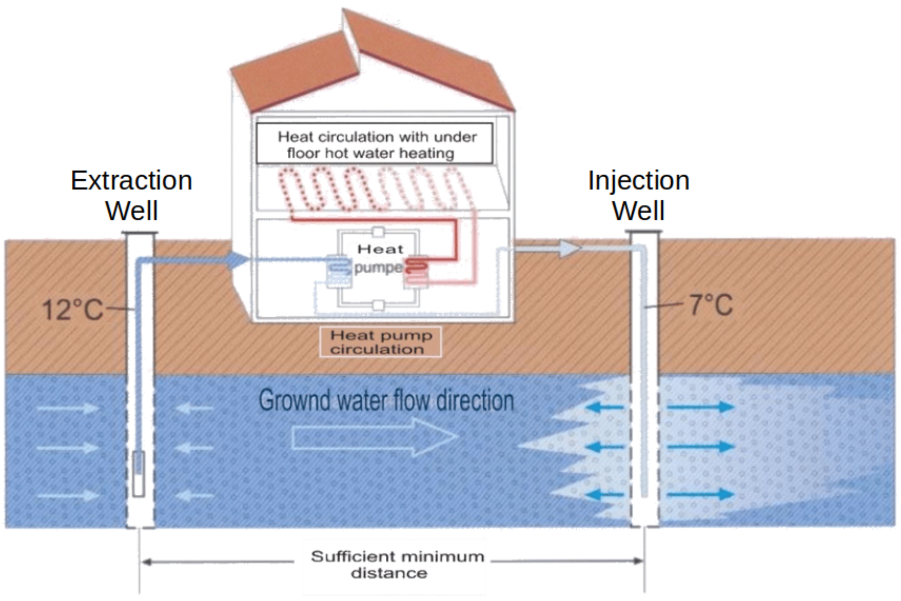

Heat circulation with under floor hot water heating
Extraction Well
Injection Well
Heat pumpe
12°C
7°C
Heat pump circulation
Grownd water flow direction
Sufficient minimum distance

FIGURE 5.2 Open loop groundwater heat pump with the resulting thermal plume. The extracted groundwater (extraction well) is run through a heat exchanger, which causes an increase or decrease of the extracted water temperature. The flow of groundwater causes a plume to develop that stretches downstream of the injection well. Image modified from [Pav16].

and reinjected at a lower temperature into the aquifer through a single injection well or a set of injection wells (right well in Figure 5.2). The opposite happens in cooling mode, where relatively cool water (once again, the water is cool relative to the building) is extracted, and warm water is reinjected back into the aquifer. This explains why GWHPs used for heating help to alleviate the SUHI effect and why cooling exacerbates the effect. The groundwater movement pulls the reinjected water with it. As the reinjected water is at a different temperature than the rest of the aquifer, this causes a thermal plume to develop and propagates downstream from the GWHP. Care must also be taken that the injection and extraction wells are sufficiently far away from each other to avoid the injected water being recycled back into the extraction well, an effect known as thermal recycling.

As energy is only transferred between the water and the building, heat pumps are highly efficient means for heating and cooling. The efficiency is a major factor determining the operating costs of a GWHP and is determined by the coefficient of performance (COP), which is the dimensionless ratio between the heating capacity Q_{hp} (how much heat is transferred) and the electrical power input P_{hp}. The larger the COP, the more units of energy are transferred per unit of energy provided to the GWHP. Several factors influence a GWHP COP, such as the groundwater extraction temperature, groundwater level and part load behaviour.

There are two main approaches for determining the COP using either the ideal Carnot cycle to determine the theoretical maximum COP [Con19], or using a quadratic function to model scaling functions for the theoretical COP. A detailed explanation of the various methods can be found in Halilovic et. al. [Hal22]. Using the quadratic function, the COP is defined as

$$COP = f_{corr}\left(9.97 - 0.2 \cdot \Delta T + 0.0012 \cdot \Delta T^2\right), \tag{5.1}$$

where ΔT is the temperature difference between the extraction well temperature and application temperature (such as the building temperature), and $f_{corr} = 0.85$ is a scaling factor to account for system losses based on field measurements. Therefore, the extraction well temperature from the subsurface aquifer directly affects the COP of each GWHP throughout a city.

5.2 Simulation Based Infrastructure Planning

Ensuring that all GWHPs within an urban environment are operating efficiently is a difficult task. With the advancement in numerical methods for groundwater simulation and optimisation solver, utilising software specifically designed for the task seems natural. In the following section, we motivate the need for such a software tool.

5.2.1 Motivation – Interference of GWHPs

Using GWHPs throughout a city has a massive potential to reduce the dependence on fossil fuel based sources for space heating, while simultaneously reducing the SUHI effect. However, a thermal plume develops from each injection well when in operation, propagates downstream, and potentially interferes with other GWHPs. A schematic of several interacting GWHPs in a small neighbourhood is shown in Figure 5.3, where the groundwater is flowing from left to right. Each house has an extraction well, an injection well and a thermal plume, which is a volume of water in the subsurface aquifer where the temperature is higher or lower than the background groundwater temperature, indicated by the blue ellipse shapes. The plume propagates downstream according to the groundwater flow direction and sometimes reaches another GWHP's extraction well. This changes the extraction temperature at the downstream GWHP and in most cases reduces the COP (efficiency) of the downstream system.

Previous research has attempted to solve similar problems before. Attard et. al. [Att20] developed a concept for determining the interference between GWHP systems by determining the probability of interference of the systems. Garcia-Gil et. al. [Gar20] developed a numerical groundwater simulation model to evaluate the impact of nested GWHP systems, but without utilising an energy optimisation solver. Similarly, Meng et. al. [Men19] evaluated the thermal impacts of shallow GWHP on a neighbourhood scale, once again without any form of energy infrastructure optimisation. Work including some form of energy infrastructure optimisation include that of Beck et. al. [Bec13] and De Paly et. al. [De 12], where either the usage or placement of GWHPs was altered. However, an analytical formulation of the heat transfer through

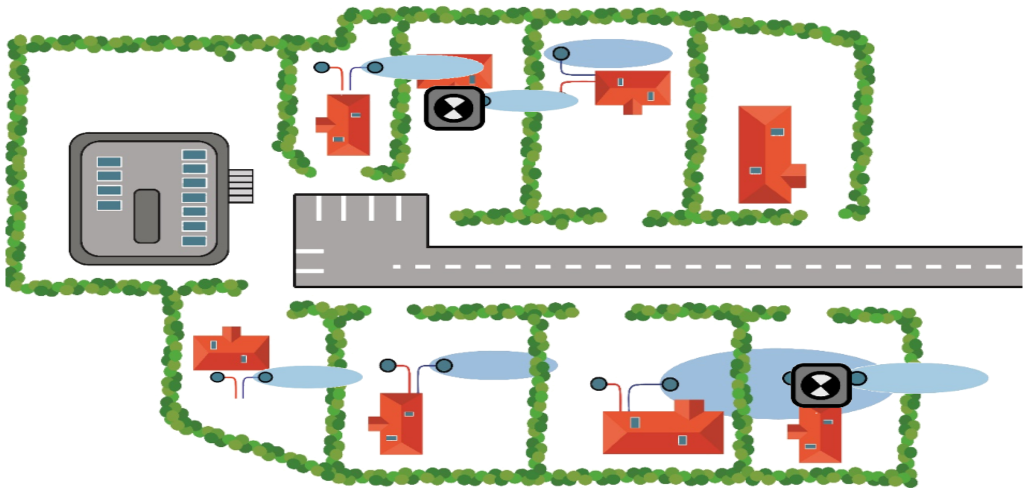


FIGURE 5.3 Thermal plumes generated from multiple groundwater heat pumps operating in a small neighbourhood. The groundwater flows from left to right, causing the plumes (coloured ellipses) to propagate downstream and potentially interfere with downstream systems (interference is designated by the ⊗ symbol).

the subsurface was used, enabling 1000's of optimisation iterations.

To carefully manage groundwater as a resource and, furthermore, to optimise the use of the shallow aquifer, detailed numerical simulations of the subsurface domain with all functioning GWHPs are required. These simulations have to cover as large a domain as possible to be able to correctly predict all positive (e.g., between a heating and a nearby cooling system) or negative (e.g., between two neighbouring heating systems) interactions between spatially distributed geothermal infrastructure components. In addition, the use of heating and cooling varies seasonally. Therefore, the simulation must cover a large time span e.g. a whole year or even several years. A building that requires space heating might only have favourable groundwater conditions during the hot summer months, and a different source of space heating might be required. An energy infrastructure optimisation solver needs to determine where GWHPs should be placed, how much each should be used and at what time of the year, to maximise the efficiency of all GWHPs interacting with the shallow aquifer.

5.2.2 Coupling procedure

The GEO.KW project aims to develop the necessary numerical groundwater simulation (NGS) and energy infrastructure optimisation (EIO), and connect them to form a tightly coupled system. The state-of-the-art at the beginning of this project assumed that the groundwater temperatures are constant for EIO [Ruh19]. However, when many GWHPs interact with one another through a single aquifer, this assumption is rarely true. The coupling idea can be divided into multiple steps.

- First, the EIO accepts data from the NGS (for example, the groundwater temperatures at each GWHP location), and, based on these temperatures, determines where GWHPs should be added to the city to optimally utilise the aquifer.
- A new NGS is performed with the updated set of GWHPs, and new subsurface flow and temperature field is obtained.
- This procedure is repeated until the EIO solution does not change any more.
- This procedure is similar to the implicit-coupling scheme for multi-physics simulations described in Section 2.2.

The tight coupling of the solvers is required to ensure that practical and legal constraints are met along with finding the optimal solution, such as minimum and maximum groundwater pressures or temperatures within drinking water sites. A detailed understanding of each solver is required to understand how the coupling can be implemented for each solver. Therefore, the sections below describe the theoretical and technical details of the individual components EIO and NGS.

5.2.3 GEO.KW Project Partners

The GEO.KW project required the collaborative effort between multiple departments, including:

- Chair of Hydrogeology, Technical University of Munich (TUM HYD),
- Chair of Renewable Energy Systems, Technical University of Munich (TUM ENS),
- SuperMUC-NG, Leibniz Research Center, Munich (LRZ).

The groundwater simulation models were developed at TUM HYD, whereas the energy infrastructure models were developed at TUM ENS. The author collaborated with all project partners at various stages throughout the development work described in this chapter, which is, thus, not the work of the author alone.

5.3 Shallow Geothermal Simulation

The shallow subsurface beneath a city is a complex, heterogeneous environment in constant interaction with the subsurface built environment of building basements, heat pumps, observation wells, tunnels or culverts. Modelling this environment is no simple task. Various complexities, including the shape of the surface terrain and physical structures, calibrated groundwater parameters and realistic boundary conditions are all required to achieve reasonable results. Therefore, selecting a simulation software that can implement all the necessary features, which is scalable on high-performance computing (HPC) systems, and that is either open-source or have extensive API capabilities to enable coupling with an external solver, is critical to the success of any simulation coupling task.

Within this project, PFLOTRAN[2] [Ham14] was selected to perform the numerical groundwater simulations. PFLOTRAN is an open-source, massively parallel, multi-phase subsurface flow simulation software that has the required numerical models allowing for the addition of GWHPs via appropriately defined boundary conditions. PFLOTRAN accepts unstructured meshes required for the complex surface terrain and subsurface structures and allows for heterogeneous groundwater parameters to be used.

5.3.1 Subsurface Modelling

PFLOTRAN supports a wide variety of subsurface flow models. To model GWHPs, we require a model where the groundwater can be extracted out of the domain, and subsequently reinjected into the domain. The groundwater temperature is an important property required for the optimisation procedure described above. Therefore, we require a groundwater model that can model the movement of the subsurface due to the application of suitable boundary conditions and can model the temperature profile of the subsurface aquifer. Below we briefly define the subsurface modelling and implementation in PFLOTRAN. However, a detailed description can be found at `https://documentation.pflotran.org/`

The subsurface is defined in a single space-time domain $\Omega_P \times [0,T] \in \boldsymbol{R}^d \times \boldsymbol{R}$ $(d \in 1,2,3)$. Beginning with the solution for a single phase, variably saturated, isothermal system, the mass conservation equation is defined as

$$\frac{\delta}{\delta t}(\varphi s \eta) + \nabla \cdot (\eta \boldsymbol{q}) = Q_w, \tag{5.2}$$

where φ is the porosity, s is the saturation ratio $[m^3 m^{-3}]$, η is the molar density of water $[kmo\ell m^{-3}]$ and Q_w is the source/sink term in $[kmo\ell \cdot m^{-3} \cdot s^{-1}]$. The Darcy velocity $\boldsymbol{q} = (q_x, q_y, q_z)^T$ in $[m \cdot s^{-1}]$ is defined as

$$\boldsymbol{q} = -\frac{\boldsymbol{K}(s)}{\mu}\nabla(P - \rho g z), \tag{5.3}$$

with $\boldsymbol{K}(s)$ is the relative permeability field in $[m^2]$, μ is the viscosity $[Pa{\cdot}s]$, P is the subsurface water pressure $[Pa]$, g is the gravitational constant $[m \cdot s^{-2}]$, and z the relative reference height $[m]$. The relative permeability field for a heterogeneous system is given be the 3-dimensional tensor

$$\boldsymbol{K} = \begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{bmatrix},$$

The addition of the thermal transport equations within the subsurface model is governed by the conservation of energy

[2] `https://www.pflotran.org/index.html`

$$\frac{\delta}{\delta t}\left(\varphi s \eta U + (1-\varphi)\rho_p c_p T\right) + \nabla \cdot (\eta \boldsymbol{q} H - \kappa \nabla T) = Q_e, \tag{5.4}$$

where ρ_r is the rock density, c_p and κ are the heat capacity and thermal conductivity of the porous medium – fluid mixture respectively, Q_e is the energy source/sink term, and U and H are the internal energy and the enthalpy of the water, respectively. They are related by the thermodynamic expression

$$U = H - \frac{P}{\eta}. \tag{5.5}$$

The initial conditions for the subsurface model are defined in the domain Ω_P and on the boundary $\delta\Omega_P$ as

$$P = P_0 \quad \text{in} \quad \Omega_P,$$

$$\boldsymbol{K}\nabla P \cdot \boldsymbol{n} = \boldsymbol{q}_0 \quad \text{on} \quad \delta\Omega_P,$$

where a uniform pressure is supplied throughout the domain and a pressure gradient is specified at each boundary of the domain, inducing a Darcy velocity $\boldsymbol{q}$ on $\delta\Omega_P$. A GWHP can be modelled by specifying a heat flux Q_e or by injecting a mass flow rate Q_w at a specified temperature. Within this work, specifying the mass flow rate and injection temperature is necessary as the injection temperature is always dependent on the extraction well temperature (see Section 6.1.1).

5.3.2 Domain Discretisation

Generating an accurate representation of the subsurface domain to run in PFLOTRAN is comprised of a number of steps. First, an accurate geometrical representation of the surface and subsurface, as well as the physical build environment, is required. Secondly, a mesh is constructed from the domain, such as a finite volume mesh, that can be used within a numerical subsurface simulation. The generation of the PFLOTRAN model was performed by the Chair of Hydrogeology at the Technical University of Munich. The parameter estimation procedure in Section 5.3.3 was a collaborative effort by the Chair of Hydrogeology, the LRZ and the author.

Geometry construction

The initial computer aided design (CAD) model was built within Salome[3], an open-source numerical simulation tool. Salome is a numerical simulation interface software that can generate the geometrical domain and the internal mesh for classical mesh-based solvers. The key components must be accounted for in the domain includes the quaternary and tertiary layers, surface water bodies, culvert systems and GWHP wells. Salome offers a Python interface, allowing for

[3]`https://www.salome-platform.org/`

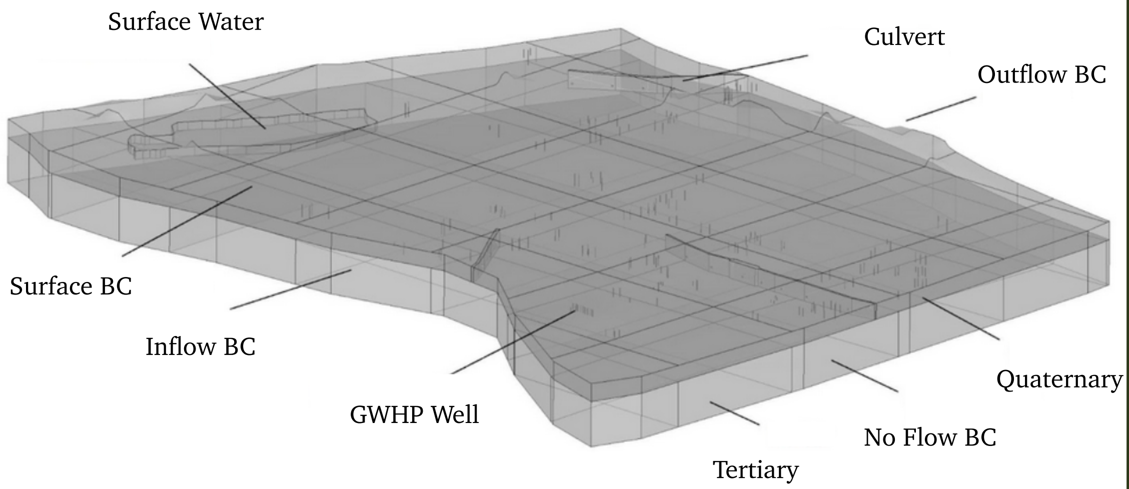


FIGURE 5.4 Shallow subsurface flow simulation: an exemplary domain with subsurface culverts, heat pump wells, surface water bodies and boundary conditions displayed. A pressure gradient boundary condition is specified at the inflow and outflow boundary conditions (BC), with no flow through the side boundaries. All domain features are stored within a database, that can be called to automate the domain generation and mesh creation steps.

the geometry generation step to be automated. Geometrical features (such as GWHP sizes and locations) are read from a database of all geometrical information and included in the geometry preparation step. An example of the domain is shown in Figure 5.4, indicating the regions of boundary conditions (BC), quaternary and tertiary layers, and subsurface infrastructure such as culverts and GWHP wells.

The geometry was built by specifying the geological boundaries of the model, accounting for the boundary between the quaternary and tertiary layers. The digital terrain model DGM2 was used to model the surface terrain, and the quaternary-tertiary layer boundary was interpolated using data from the GEPO project[4]. The irregular surface terrain was imported a set of raster pixels in 3 dimensions, and the surface was created by fitting B-splines through the pixels.

Domain Meshing

Creating a finite volume mesh from the domain geometry is required to perform the numerical groundwater simulations. Once again, Salome was used to create an unstructured tetrahedral mesh of the computational domain. Tetrahedral meshes are beneficial for highly irregular domains and can adhere to the form of a complex shapes. However, this comes at the cost of generating many finite volume elements, and therefore, a high computational cost. Secondly, the mesh quality may also suffer due to the non-orthogonality and high aspect ratio of the elements. Therefore, the computational domain was converted into a polyhedral mesh. Polyhedral meshes offer a

[4]`https://www.cee.ed.tum.de/en/hydro/projects/gepo/`

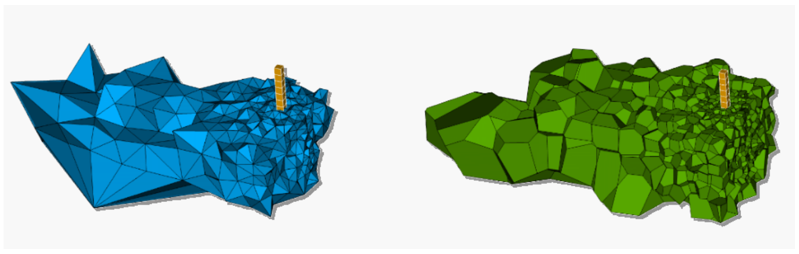


FIGURE 5.5 Shallow subsurface flow simulation: tetrahedral mesh (left) and polyhedral mesh (right). The tetrahedral mesh allows for meshing of complex geometries, but often comes at the cost of mesh quality and a large number of elements. The polyhedral mesh is generated by fusing adjacent tetrahedral elements together to form a single element. This improves the mesh quality by improving the orthogonality and aspect ratio of the elements, and reduces the number of elements in the mesh.

better mesh quality with a reduced number of elements that reduces the computational runtime. An example of the two mesh types is shown in Figure 5.5. The tetrahedral mesh was converted to a dual voronoi graph from the Geogram library[5], using the SALOME-Voronoi interface[6]. The polyhedral mesh is generated by fusing tetrahedral elements together, which is one of the reasons why the number of elements can be drastically reduced.

Many features exist within the subsurface that require a mesh of sufficient resolution to resolve. Figure 5.6 displays the surface terrain of the with the tetrahedral mesh (background), a refined mesh resolution around a GWHP well (top right) and the polyhedral mesh at the intersection of the quaternary and tertiary layers, with a subsurface structure placed in the quaternary layer (bottom left). The high mesh resolution around each GWHP results in a large mesh when hundreds or thousands of GWHPs are placed within a single PFLOTRAN domain. A viscous layer (green) is defined around subsurface infrastructure to force the voronoi meshing procedure to adhere to the shape of the structures. Similarly, a flat interface is defined between the quaternary layer (pink) and tertiary layer (light blue) for the meshing software adheres to the interface layer.

5.3.3 Parameter Estimation

Estimating the groundwater parameters, also commonly referred to as calibration, is vital to determine the correct groundwater properties. Parameter estimation involves varying the input parameter field in the numerical model, such that real-world measurement data are replicated in the simulation. This is usually performed to obtain the correct properties such as hydraulic permeability, porosity and thermal conductivity, which are usually represented by heterogeneous fields throughout the domain. An example of a time-series real-world datasets, obtained from monitoring facilities throughout the city, are shown in Figure 5.7.

Once the PFLOTRAN model domain has been constructed, meshed, and the boundary condi-

[5]`https://github.com/BrunoLevy/geogram`
[6]`https://github.com/MoiseRousseau/SALOME-Voronoi`

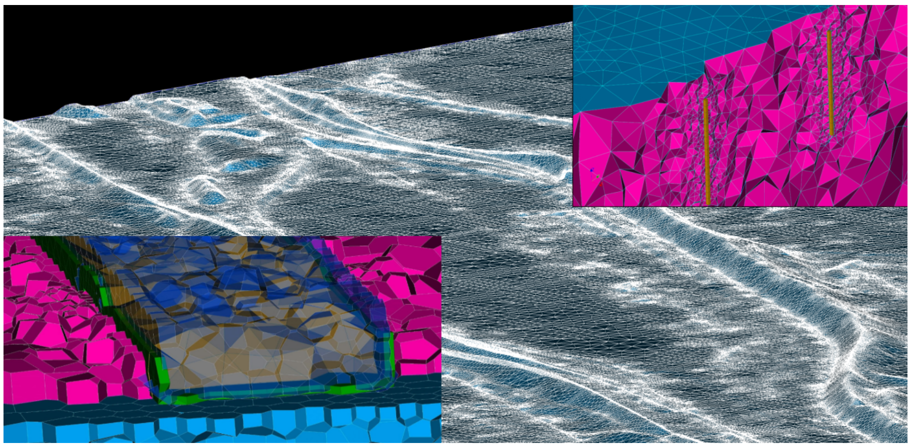


FIGURE 5.6 Shallow subsurface flow simulation: groundwater simulation mesh of the surface terrain, heat pumps and subsurface features. A tetrahedral mesh showing the refinement of the mesh around each well in the domain (top right), surface terrain features that are captured in the domain geometry (background), and subsurface features showing the interface between the quaternary and tertiary layers with a polyhedral mesh (bottom left).

tions have been applied; the final step is to calibrate the model. The parameter estimation software PEST++[7] [Doh15] was used to calibrate the PFLOTRAN domain. PEST++ utilises the method of pilot points to parameterise the domain, shown in Figure 5.8. This specifies the groundwater property value at each pilot point (solid dots in Figure 5.8) and maps the values from the points to the entire PFLOTRAN mesh to provide a heterogeneous field. The parameter estimation procedure begins by creating an initial parameter field (uniform, random or preferred values can be specified at each pilot point) applied to the pilot points, mapped to the PFLOTRAN mesh and the initial simulation is run. The observation values (cross in Figure 5.8) at the identical locations of the real-world measurements are extracted from the simulation, and compared to the real-world measurements. Based on the observation values, PEST++ selects new groundwater property values at each pilot point. The values at the pilot points are again mapped to each element in the PFLOTRAN domain, and a new groundwater property field is generated. The PFLOTRAN simulation is re-run with the new parameter field, after which the observation values are again extracted. This process is repeated until the simulation observation values match the real-world data.

There are multiple factors that greatly influence the parameter estimation results: *(i)* the parameter estimation technique and *(ii)*, the mapping technique from the pilot points to the PFLOTRAN mesh. Firstly, Tikhonov regularisation was selected as the parameter estimation method. Tikhonov regularisation is often applied to ill-posed problems, which is commonly found in problems with a large number of parameters. The generalised Tikhonov regularisation can be defined as the minimisation of the simulation output that matches the real-world measurements,

[7]https://pesthomepage.org/

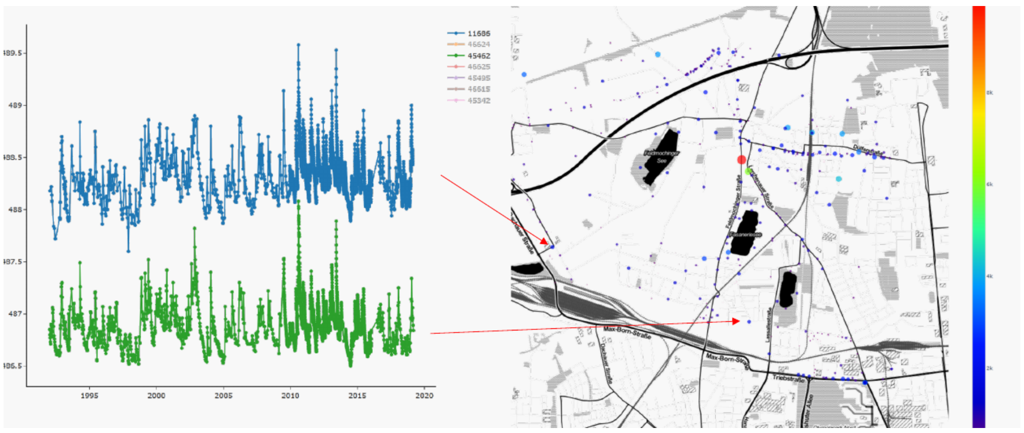


FIGURE 5.7 Shallow subsurface flow simulation: time series data of real-world observations that are used to calibrate groundwater models. Once suitable boundary conditions have been added to the model, the parameter estimation procedure determines the groundwater properties, such as hydraulic permeability of thermal conductivity. Typical real-world measurements include hydraulic head level, temperature or groundwater velocity.

while also staying as close to some known input values

$$\min \rightarrow \frac{1}{n}\sum_{i=0}^{n} ||\boldsymbol{y}_i - \boldsymbol{y}_{obs}||^2 + \sum_{i=0}^{n} \lambda ||\boldsymbol{y}_i - \boldsymbol{y}_{pref}||^2, \tag{5.6}$$

where n is the total number of measurements, $\boldsymbol{y}_{obs}$ are the real-world measurement values, $\boldsymbol{y}_i = f(\boldsymbol{x}_i)$ are the observation output values from the simulation at iteration i, λ is a scaling factor specifying the strength of the regularisation between minimising the real-world measurements versus matching the preferred parameter values of $\boldsymbol{y}_{pref}$. In this work, we do not evaluate the ability of other parameter estimation tools in PEST++, and instead rely on industry standard techniques to calibrate the domain.

Secondly, the radial basis function mapping using the thin-plate-spline basis function, was used to map the parameter values from N_{pp} pilot point locations to N_{mesh} mesh elements. As $N_{pp} \ll N_{mesh}$, the cost of solving the interpolation field, Equation 2.37, is almost negligible compared to a single PFLOTRAN simulation solve. However, the size of N_{mesh} can be large (where $N_{mesh} > 10^6$ and $N_{pp} < 10^2$ is common), and the available computing memory is a limiting factor when mapping the values to the PFLOTRAN mesh. Therefore, the RBF mapping technique was parallelised, where solving Equation 2.37 was repeated on each computing rank. The PFLOTRAN mesh was decomposed across all computing ranks and the interpolation step of Equation 2.36 was limited to the local mesh on each rank only. This reduced the size of matrix $\boldsymbol{C}$ in Equation 2.36 and reduced the interpolation runtime. Finally, all information was sent to the main rank to save the parameter file, *parameter.h5*, to be read by PFLOTRAN. The parallel RBF mapping procedure pseudo-code is shown in Algorithm 5.1.

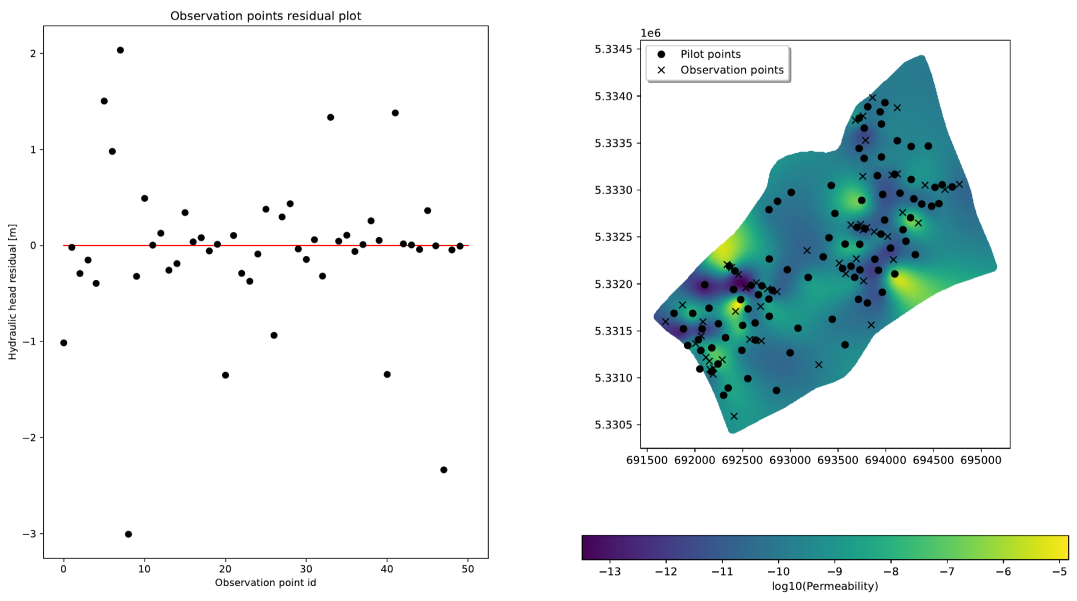


FIGURE 5.8 Shallow subsurface flow simulation: pilot point parameter estimation procedure using PEST++, with the final hydraulic head error (left) at each observation point (crosses, right) and the pilot point locations (solid dots) with the final permeability solution (right). The parameter estimation procedure was validated using an artificial test case. An artificial permeability field was generated to produce a set of "real-world" measurements at the observation points. The calibration procedure was performed to recreate the permeability field using the observation point values as the real-world measurements.

```
Input: input mesh $\boldsymbol{x}_{\Gamma_{pp}}$, output mesh $\boldsymbol{x}_{\Gamma_{mesh}}$, parameter value at each pilot point $y_{pp}$
Output: groundwater parameter values in PFLOTRAN mesh $y_{mesh}$
Copy mesh $\boldsymbol{x}_{\Gamma_{pp}}$ and values $y_{pp}$ onto each compute rank
Load output mesh $\boldsymbol{x}_{\Gamma_{mesh}}$ onto main rank only
Divide the output mesh equally over
The number of output vertices per rank is then $\lceil \frac{N_{\Gamma_{mesh}}}{N_{ranks}} \rceil$
Solver RBF interpolation on each rank
Interpolate solution to vertices on local rank only
Combine results onto main rank
Save results into H5 PFLOTRAN input file.
```

ALGORITHM 5.1 Pseudo-code for the parallelised pilot point RBF mapping method. The pilot point vertices are copied on all ranks, whereas the PFLOTRAN mesh is divided equally across all ranks.

5.3.4 Summary of Geothermal Modelling

This section explained the features required to perform accurate numerical groundwater simulations. The numerical model was defined, briefly explaining the governing equations used to resolve the subsurface temperature and pressure fields that are required for partitioned simulation-optimisation coupling. Next, the domain generation and meshing technique was discussed, showing the detail accounted for in the model to attain accurate simulation results. Finally, a brief explanation of the parameter estimation procedure was discussed, defining the type of parameter estimation and mapping methods. Defining all aspects of the numerical groundwater simulation, in detail, is beyond the scope of this work. However, enough detail was provided to ensure that sufficient effort was taken to ensure that the simulation-optimisation procedure is as accurate as possible.

5.4 Energy Infrastructure Modelling

The combination of groundwater simulation and energy infrastructure optimisation has the potential to be a powerful tool for planning of GWHP expansion. This requires that the infrastructure optimisation model can account for already existing infrastructure and potentially thousands of GWHPs throughout the city. In the following section, we provide an introduction in infrastructure capacity planning and optimisation, along with a description of the software used within the GEO.KW project.

5.4.1 Energy Infrastructure Networks

In the simplest of terms, infrastructure is defined as a set of services, facilities and systems that serve either an area, city or country, and allows its economy to function. The work of Buhr [Buh03] separates infrastructure into three main types: institutional, personal and material. The material infrastructure definition is the most commonly understood, which represents capital goods, such as buildings, installations and equipment for education, health care, transportation, water sanitation and water sewage works, energy, electricity and management of natural resources. Institutional infrastructure refers to rules and procedures typically provided by public institutions, whereas personal infrastructure refers to people, specifically the education and skills within a community.

Within the context of the GEO.KW project, infrastructure takes on the material definition, such as the physical built environment. Specifically, the various components of the energy infrastructure network are examined and modelled. This comprises all equipment that is used for the storage of materials, for conversion systems and energy distribution networks.

In order to provide heating and cooling services throughout a city, a variety of different methods, each with their own dedicated infrastructure, are used.

Electricity:

At the heart of our modern society lies electricity, powering everything from our lights at home to cars and trains transporting us around the world. Current large scale energy transmission uses alternating current (AC), which allows for high voltage low current transmission with low thermal losses. Specialised electrical equipment can provide space heating and cooling for households and large buildings, where the running cost is directly dependent on the cost of electricity. However, CO_2 free space heating and cooling is possible if electricity is generated from renewable sources.

Natural Gas:

Natural gas has been widely used since the start of the 19th century and is a common energy source for heating of households and buildings within Germany [Leu09]. Initially, gas was used in limited quantities for cooking and water heating until the 20th century when many safety issues were addressed. Space heating took off with advances made in designing more efficient radiators. In the 21st century, gas networks were built and operated by large energy companies, which manage the flow of gas through the network to ensure the efficient and safe usage of gas networks.

District Heating & Cooling (DH&C):

District heating generates heat at a centralised location, transfers the heat to a working fluid, and distributes the warm fluid to the end user. Similarly, district cooling involves cooling down a working fluid, and distributing the cool fluid instead. The working fluid is transported throughout a region through highly insulated pipes to reduce thermal losses. At the end user, the working fluid is run through a heat exchanger to either heat up or cool down a building, after which the fluid is transported back to the central location to form a closed system. DH&C has the benefit of having all heating and cooling equipment situated at a single location, making for easier accessibility and maintenance. However, thermal stresses within the pipe network are the main cause for ageing and wear.

In addition to the above infrastructure components to provide heating and cooling, shallow geothermal heat pumps, described in Section 5.1.3, are promising alternatives that are being widely used.

5.4.2 Energy Modelling Components

There are many ways to provide space heating and cooling for buildings other than using GWHPs. To build an energy infrastructure optimisation model to accurately select GWHP locations and loads, the various components that are relevant for infrastructure planning must be defined. It is not sufficient for the model to only maximise the use of GWHPs everywhere, regardless of the cost. By integrating the GWHPs into the energy optimisation model, the impact the GWHPs have on the current infrastructure can be accounted for. The definition of the energy infrastructure optimisation method presented below is derived from Dorfner [Dor15].

Commodities

The first aspect to understand about the energy infrastructure model are commodities. Commodities are variables that represent a form of mass or energy, i.e., energy from solar rays per m^2, energy content from a kilogram of natural gas, or kilograms of CO_2. The set of all commodities C is divided into four main types: stock commodities C_{st}, supply intermittent commodities C_{sup}, demand commodities C_{dem} and environmental commodities C_{env}. Stock commodities are vast enough to the used almost unrestrictedly, such as coal. Supply intermittent commodities are those that vary due to external factors, such as wind or solar power. Demand commodities are those that are required by other conversion processes or load curves, such as electricity or heat (a GWHP requires electricity to run). Finally, environmental commodities are products of the conversion processes, such as CO_2 or solid waste (ash from a coal power plant).

Demand Load Curves

The demand is represented by a time-dependent load curve and is the requirement for a specific commodity over a defined time period $d_c(t)$. An example is the space heating load of a building for each day over the winter months. Load curves can be continuous functions or can be specified at discrete time points. For energy infrastructure optimisation, is common to convert continuous load curves into discrete values.

Conversion Process

Fundamental to the optimisation process is the conversion of commodities to demands. This can be thought of as black-box that converts incoming power flow from commodities ϵ_{pt}^{in} to an outgoing power flow ϵ_{pt}^{out} (as the demand is measured in terms of energy), at a specific point in time. All conversion processes have losses that are accounted by a scaling factor, such that $\epsilon_{pt}^{out} = e_p \epsilon_{pt}^{in}$. Each conversion process is also limited between a minimum and maximum capacity that it is able to provide, defined by $\epsilon_{pt}^{out} \geq \kappa_{p,min}$ and $\epsilon_{pt}^{out} \leq \kappa_{p,max}$.

An example of a conversion process is that of a GWHP. It receives inputs in the form of electricity and supplies a heat power output that can be used to meet a demand. Another example is a gas turbine. The gas commodity flows into the gas turbine (conversion process), and electricity, heat and CO_2 are output commodities from the conversion process.

Space-Time Decomposition

The solution of the optimisation problem is defined over both space and time. In the optimisation model developed at TUM-ENS, the domain comprises of discrete points in space, called vertices or sites. These sites are connected by undirected edges or directed arcs to form a graph. The sites can be interpreted as houses, buildings, districts, cities or countries and are defined as locations where commodities and/or a demand exists. The edges indicate transport paths for commodities. Conversion processes are assigned to sites as required.

At each site exists a time-dependent demand. Time is discretised using discrete points $\boldsymbol{T} = [t_0, t_1, t_2, \ldots, t_N]$ corresponding to the demand from the discretised load curve $d_c(t)$. The size of the time step Δt is dependent on the optimisation problem to be solved. When optimising household electricity demand, there is significant variation through the day, for example due to the morning and evening peaks. Therefore, a small time step of $\Delta t = 60$ minutes or $\Delta t = 15$ minutes is chosen. However, if only considering the usage of shallow GWHPs, the effects are slow moving, and a time step of $\Delta t = 1$ day is better suited.

Miscellaneous

There are various other components required to perform the energy infrastructure optimisation beyond what is described in this work. Additional components such as the transmission of commodities between sites and conversion processes, and storage of commodities at sites. Certain commodities can be stored, e.g., electricity in a battery, potential energy in a pumped-storage facility, or heat within a molten salt storage facility. The commodities generated in one site can be used to satisfy the demand of another site, given that a suitable means of transmission exists. For example, one site may have gas commodities to generate electricity, but little electricity demand, but this can be transferred to another site where there is electricity demand but no generation capacity. The reader is referred to [Dor20] for detailed information on energy infrastructure networks.

Cost

Each commodity and conversion process has an associated cost ζ. Each conversion process as described above incurs an investment cost ζ_{inv} to build the required infrastructure, a fixed cost ζ_{fix} for continuously using a process (such as land rental for a solar farm), and variable costs ζ_{var} that may be process specific (such as fluctuating maintenance expenses if a process is used frequently). Similarly, commodities may sometimes incur a cost of usage, fuel costs such as gas or coal ζ_{fuel}. Other commodities are free, such as wind or solar radiation, however the infrastructure and maintenance costs to harness them is not and accounted for in the conversion process cost. A single-input, single-output example is shown in Figure 5.9, showing the function of two GWHP at two sites. The solar PV panel (conversion process) generates electricity (commodity) from the solar resource (commodity) and powers a heat pump (conversion process).

5.4.3 Energy Infrastructure Optimisation

As described in the planning and optimisation tool definition in Section 5.2, improving the use of shallow GWHPs throughout a city requires optimising the usage (what is the mass flow rate of groundwater through the GWHP) and placement (where it is) of GWHPs. Optimising the usage and placement requires for multiple factors to be considered.

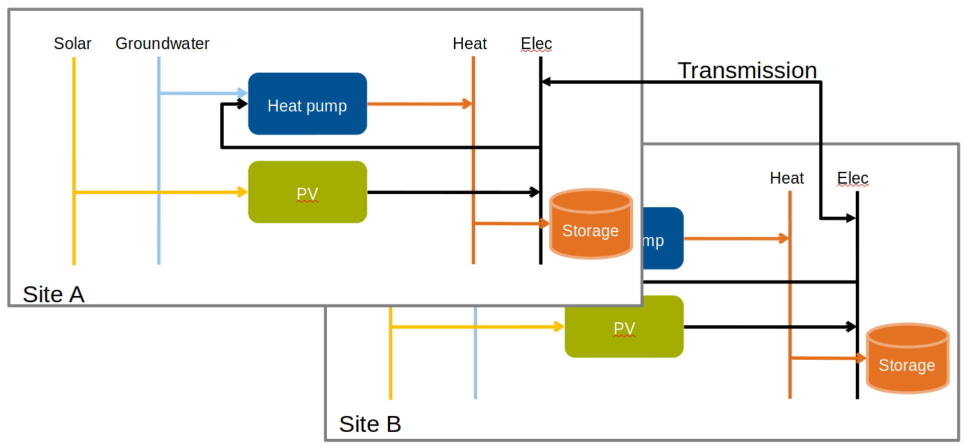


FIGURE 5.9 Single-input, single-output example of two conversion processes on two sites, A and B, with transmission. The solar panels require sunlight (solar) to generate electricity. This electricity can power a GWHP (heat pump), that also requires groundwater to generate heat. The heat can be stored, however not heat transmission capacity exists.

- The first, and most obvious, is that sufficient groundwater should be available (the capacity) at ideal temperatures and pressures within the subsurface (obtained from the numerical groundwater simulations in Section 5.3).
- Secondly, the actual demand of heating and cooling within specific areas needs to be accounted for (the demand).
- Thirdly, factors affecting the cost must be considered, such as installation costs for drilling, investment costs and operational costs.
- Finally, the aim of the optimisation must be specified, which could be to reduce cost or CO_2 etc. (the objective function).

In order to perform an optimisation, it is necessary to know what the desired solution should aim to achieve. The two mentioned objectives, costs and CO_2 would influence the results in different ways. If the objective function is to reduce the total cost, then only the cheapest commodities and conversion processes would be selected in the optimisation process. If minimising the CO_2 output, then most non-fossil fuel based sources would be selected. In the following subsection, we explain how the objective function is formulated in the cost minimisation case.

Objective Function

Some areas might be too costly to drill at to install a GWHP, or the demand may be too low. Regardless of how suitable a GWHP might be to install in that area, it may not be economically viable if reducing costs is the objective. Or, if reducing CO_2 is the objective, the cost of installation may not be a setback. Therefore, the choice of objective function plays a large role in the quality of results. First, the standard form for a non-linear optimisation problem is defined as

$$\min_{\boldsymbol{x}\in\boldsymbol{\chi}} f(\boldsymbol{x}), \quad \text{such that} \quad g(\boldsymbol{x}) \leq 0, \quad h(\boldsymbol{x}) = 0, \tag{5.7}$$

where $f : R^n \mapsto R$ is the objective function, $\boldsymbol{\chi} \subseteq R^n$ is the space of all possible solutions and $\boldsymbol{x}$ is the optimal solution. Additional constraints are provided by functions $g : R^n \mapsto R^m$ of m inequality constrains, and $h : R^n \mapsto R^p$ of p equality constraints. The set of feasible solutions $\mathscr{F}$ is the subset of $\boldsymbol{\chi}$ that satisfies all constraints,

$$\mathscr{F} = \{\boldsymbol{x} \in \boldsymbol{\chi} \mid g(\boldsymbol{x}) \leq 0 \wedge h(\boldsymbol{x}) = 0\}$$

There are various methods capable of solving the optimisation problem of Equation 5.7. Linear programming (LP) converts the optimisation problem to a set of continuous, linear functions. The advantage of LP optimisation is that it scales well to large systems and there is a wide variety of software solvers available to solve the respective system. However, it is limited to linear problems only (or problems that can be converted to a linear representation), and therefore, difficult to represent complex interactions.

Integer programming (IP) converts the solution space $\boldsymbol{\chi}$ to a discrete space. Typical applications include production planning, scheduling and routing problems, where solution variables cannot be defined continuously. Similar to LP, IP scales well to large systems and complex interactions can be modelled. However, IP suffers from bad worst-case complexity to find a solution.

Therefore, continuing with the LP optimisation method, the optimisation formulation can be rewritten

$$\min_{\boldsymbol{x}\in\boldsymbol{\chi}} \boldsymbol{c}^T\boldsymbol{x}, \quad \text{such that} \quad \boldsymbol{A}\boldsymbol{x} \leq b, \quad \boldsymbol{x} \geq 0, \tag{5.8}$$

Software Implementation: urbs

The energy infrastructure optimisation software used in this thesis is urbs[8], initially developed by Richter [Ric04]. Solving the LP problem first requires transforming all commodities, load curves, conversion processes, cost, and every other component of the energy infrastructure model into Equation 5.8. The objective function in urbs is the total cost ζ, i.e., we aim to determine the cost-optimal use of resources $\boldsymbol{p}$ within the available capacities $\boldsymbol{\kappa}$, where a specific set of processes $\boldsymbol{\epsilon}$ and transmission between sites $\boldsymbol{\pi}$ exists, all while meeting the required demand $\boldsymbol{d}$, thereby solving for the optimal cost

$$\zeta' = \min_{\boldsymbol{p},\boldsymbol{\kappa},\boldsymbol{\epsilon},\boldsymbol{\pi},\boldsymbol{d}} \zeta \tag{5.9}$$

The minimisation problem is solved on a discrete graph, consisting of various sets. The first two sets to define are the vertices $v \in \boldsymbol{V}$ of the graph (sites) and time steps $t \in \boldsymbol{T}$. All other sets

[8]https://github.com/tum-ens/urbs

TABLE 5.1 Sets of variables used for energy network infrastructure modelling. The superscript indicates if the commodity flows into or out of a vertex/edge. The subscript indicates the process, commodity, vertex, edge or cost. For example, P^{in}_{vc} specifies a conversion process P, for commodity c, which flows in at vertex v.

Variable	Examples	Description
$v \in \boldsymbol{V}$	$v_0,\ldots,v_{N_{sites}}$	Vertices in graph
$t \in \boldsymbol{T}$	$t_0,\ldots,t_{N_{times}}$	Time steps for the optimisation solver
$e \in \boldsymbol{E}$	E^p_v, E^s_v	Directed edges incoming and outgoing from vertex v
$c \in \boldsymbol{C}$	$C_{st}, C_{sup}, C_{dem}, C_{env}$	Commodities
$p \in \boldsymbol{P}$	$P^{in}_{vc}, P^{out}_{vc}$	Conversion processes; consuming or generating commodities at vertex v
$s \in \boldsymbol{S}$	S_{vc}	Storage of commodity c at vertex v
$f \in \boldsymbol{F}$	$F^{out}_{vc}, F^{in}_{vc}$	Transmission outgoing and incoming at vertex v

defined at or between the vertices v are shown in Table 5.1.

The total cost ζ is divided into multiple costs.

$$\zeta = \zeta_{inv} + \zeta_{var} + \zeta_{fix} + \zeta_{fuel} \tag{5.10}$$

The investment cost ζ_{inv} includes the annualised investment costs for processes, transmission and storage for all new capacities $\hat{\kappa}$. The variable costs ζ_{var} vary depending on the type. Process variable cost are calculated per unit of the output ϵ^{out}_{vpt}, whereas transmission variable costs are per unit of incoming power π^{in}_{af}. Fixed costs ζ_{fix} are determined for the total capacities κ, and not only newly installed capacities. Fuel costs ζ_{fuel} sum the costs of all stock commodities. All four costs are defined in detail in Equations 5.11, 5.12, 5.13 and 5.14.

$$\zeta_{inv} = \sum_{v\in V, p\in P} \hat{\kappa}_{vp} k^{inv}_{vp} + \sum_{v\in V, s\in S} \left(\hat{\kappa}^c_{vs} k^{c,inv}_{vs} + \hat{\kappa}^p_{vs} k^{p,inv}_{vs}\right) + \sum_{a\in A, f\in F} \hat{\kappa}_{af} k^{inv}_{af}, \tag{5.11}$$

where k^{inv}_{vp}, $k^{p,inv}_{vs}$, $k^{c,inv}_{vs}$ and k^{inv}_{af} (units $[€/(MW \cdot a)]$) are the annualised process capacity, storage power, storage size and transmission capacity investment. The capacities $\kappa^{\hat{}}_{vp}$, $\kappa^{\hat{c}}_{vs}$, $\kappa^{\hat{p}}_{vs}$ and $\kappa^{\hat{}}_{af}$ (units $[MW]$) are the newly added process capacity, storage size, storage power and transmission capacity. The fixed cost

$$\zeta_{fix} = \sum_{v\in V, p\in P} \kappa_{vp} k^{fix}_{vp} + \sum_{v\in V, s\in S} \left(\kappa^c_{vs} k^{c,fix}_{vs} + \kappa^p_{vs} k^{p,fix}_{vs}\right) + \sum_{a\in A, f\in F} \kappa_{af} k^{fix}_{af}, \tag{5.12}$$

is comprised of k^{fix}_{vp}, $k^{p,fix}_{vs}$, $k^{c,fix}_{vs}$ and k^{fix}_{af} (units $[€/(MW \cdot a)]$), which are the fixed annual costs of the process capacity, storage power, storage size and transmission capacity for all capacities κ_{vp}, κ^c_{vs}, κ^p_{vs} and κ_{af} (units $[MW]$). The variable cost

(5.13)
$$\zeta_{var} = \sum_{t\in T_m}\left(\sum_{v\in V,p\in P}\epsilon^{out}_{vpt}k^{var}_{vp} + \sum_{a\in A,f\in F}\pi^{in}_{af}k^{var}_{af} + \sum_{v\in V,s\in S}\left[\epsilon^{con}_{vst}k^{c,var}_{vs} + \left(\epsilon^{in}_{vst}+\epsilon^{out}_{vst}\right)k^{p,var}_{vs}\right]\right),$$

is similarly comprised of k^{var}_{vp}, $k^{p,var}_{vs}$, $k^{c,var}_{vs}$ and k^{var}_{af} (units [€ /(MWh)]), which are the variable annual costs of the process capacity, storage power, storage size and transmission capacity. However, this is also dependent on outgoing power flow ϵ^{out}_{vpt} $[MWh/h]$, incoming transmission power π^{in}_{af} $[MWh/h]$, storage content ϵ^{con}_{vst} $[MWh]$, and difference in incoming and outgoing power ϵ^{in}_{vst} and ϵ^{out}_{vst} $[MWh/h]$.

(5.14)
$$\zeta_{fuel} = \sum_{t\in T_m}\sum_{v\in V}\sum_{c\in C_{st}}\rho_{vct}k^{fuel}_{vc}.$$

The fuel costs are the stock commodity costs k^{fuel}_{vc} [€ $/MWh$] and usage amount of each stock commodity $\rho_{vct}k^{fuel}_{vc}$ in $[MWh/h]$.

The objective function can only be solved by applying a set of suitable constraints in the LP model. There are many constraints that are required to ensure a feasible solution is found, and the reader is referred to the urbs documentation [Dor20] for all constraints available in urbs. However, the most important constraint is the commodity balance $CB(c,v,t)$, which is a function of the commodity, vertex location and time.

(5.15)
$$CB(c,v,t) = \sum_{p\in P^{out}_{vc}}\epsilon^{out}_{vpt} - \sum_{p\in P^{in}_{vc}}\epsilon^{in}_{vpt} + \sum_{a\in A^{p}_{v},f\in F^{in}_{vc}}\pi^{out}_{aft} - \sum_{a\in A^{s}_{v},f\in F^{out}_{vc}}\pi^{in}_{aft} + \sum_{s\in S_{vc}}\left(\epsilon^{out}_{vst} - \epsilon^{in}_{vst}\right).$$

which balances the power output ϵ^{out}_{vpt} and power input ϵ^{in}_{vpt}, outgoing transmission π^{out}_{aft} and incoming transmission π^{in}_{aft} power, and the change in outgoing and incoming storage power ϵ^{out}_{vst} and ϵ^{in}_{vst}, all having units MWh/h. This ensures that at each vertex v, at each time step t and each commodity c, all power inflow and outflow must be balanced. Demand satisfaction is the requirement that all demand commodities must be met by the optimisation model

(5.16)
$$d_{cvt} \leq CB(c,v,t) \quad \forall c\in C, \forall v\in V, \forall t\in T,$$

such that no demand cannot go unmet. This is solved by adding a fall back commodity that is more expensive than all other commodities. If using this fall back commodity, then the cost increases dramatically, and the minimising cost objective function should not choose this option

A GWHP is modelled within urbs by converting the electrical input of the GWHP into a heat demand via the COP, with a maximum upper capacity κ_{gwhp}. Therefore, if the price of electricity increases, or the COP decreases, the GWHP becomes a less favourable option to meet the space heating demand. The COP for each GWHP is modelled by Equation 5.1, where ΔT is the temperature difference between the extraction well temperature and the load demand curve temperature $T_{\ell dc}$. The load demand curve temperature[9] is determined independently for each GWHP, and accounts for various factors including the type of building and space heating and cooling demands.

The urbs model described above is not able to understand the influence that a GWHP has on the shallow aquifer. The capacity of the groundwater resource, based on the optimisation formulation in Section 5.4.3, is not explicitly defined, but only accounted for in the COP. There is also no constraint that accounts for the interference between GWHPs, ensuring that the GWHPs operate within the operational limits. During the optimisation process, urbs selects a subset of GWHPs that satisfy the optimal solution of the given LP problem. However, the optimal solution may select GWHPs that cause a significant amount of interference between systems due to the lack of an interference constraint. This issue can only be resolved by a suitable coupling between urbs and the groundwater flow simulation that will be presented as the main contribution of part II of this thesis in Chapter 6.

Considering that there are potentially thousands of GWHPs that can be placed within the city, and therefore, the number of combinations of GWHPs is too large to consider simulation every possible combination, a decomposed optimisation procedure was developed.

- Firstly, the optimisation problem was decomposed into many smaller, optimisation problems that are each solved independently and simultaneously.
- The urbs model does not arbitrarily select a location in space to add a GWHP. A set of locations where a GWHP can be installed is defined in urbs, and the optimisation model selects which GWHPs have a non-zero capacity, i.e., are in use. The locations are chosen due to building constraints, ease of access for GWHP installation etc.
- Each urbs model would add only one new GWHP to an already optimised solution in successive optimisation iterations. Therefore, if any interaction is found between two GWHPs in a single urbs model, the offending GWHP can be removed from the optimised solution.
- This process is repeated until there are no more valid GWHPs to add to the optimised solution for each model.

Decomposing the urbs optimisation problem into many smaller domains that are run simultaneously introduces a new problem. All urbs models interact with a single PFLOTRAN simulation domain. Therefore, if a GWHP in one urbs model influences a GWHP in another urbs model, the offending GWHP cannot easily be removed unless the urbs models communicate with one another. A fully parallel, staggered iteration approach was developed within this work to solve this problem and explained in Section 6.1.3.

[9]The urbs models and load demand curves are determined by TUM ENS.

5.5 Summary of Chapter 5

Chapter 5 introduces the need to use GWHPs to solve the SUHI problem, and to reduce the usage of fossil fuel sources for space heating needs. A novel idea was developed to coupled single physics solvers together to provide an advanced method to optimise the use of GWHP usage with a shallow aquifer. The purpose of this thesis is to build upon the knowledge of simulation software coupling, and to develop a coupling approach for the single physics solvers within the GEO.KW project. Therefore, the initial work, performed by and with the project partners, was introduced and discussed in this chapter. A thorough understanding of how each solver works was required to provide enough technical content to explain the simulation coupling in Chapter 6. However, completely describing all the work performed by the team is beyond the scope of this thesis.

6 Coupling Schemes for Partitioned Shallow Geothermal Resources Optimisation

In order to attain realistic and accurate simulation results for the GWHP layout and usage throughout a city, combining the energy infrastructure optimisation solver and the numerical groundwater simulation is paramount. This can only be achieved by tightly coupling the partitioned software components. A coupling of this nature, between an energy optimisation linear programming solver using a directed graph and a numerical groundwater simulation using a finite volume mesh, provides unique challenges. Defining the spatial coupling interface between the two mesh based solvers is common in surface coupled multi-physics problems but does not exist for the directed graph. Combined with a domain decomposition method to achieve scalability and feasibility for the large simulation and optimisation domains, various solutions were devised to solve coupling problems not seen before. Section 6.1 begins by describing boundary conditions within PFLOTRAN and urbs, which defines the information that is to be exchanged during the coupling process. Next, the coupling scheme developed in this thesis that allows for PFLOTRAN to reach a converged state before the energy system optimisation occurs, is introduced. The domain decomposition method is described in Section 6.1.3, which is required to perform the staggered coupling scheme developed in this thesis. This is followed by the parallelisation of the staggered coupling scheme and data mapping between the solvers. The implementation of the simulation-optimisation coupling is described in detail in Section 6.2. Finally, a test case is presented in Section 6.3 to prove that the coupling method works as intended.

6.1 Geothermal Optimisation Coupling Schemes

Before diving into the implementation of a new partitioned coupling scheme, a theoretical definition of the coupling needs to be defined. This section provides a complete overview of the theoretical coupling procedure between PFLOTRAN and urbs, and describes the flow of information, before discussing the software implementation in the following section.

6.1.1 Overview of Information Exchange

Before developing any new partitioned coupling simulation solver, it is necessary to first define the coupling conditions.

- What coupling-related information does each simulation software require to perform its task? Can this information be extracted from the other software?
- In what order is this information required to provide suitable coupling?
- Where is the coupling information available in a distributed memory parallelisation setting? Does it lie on a single computing rank or is it distributed across many ranks?

The aim of the coupling process is to couple the numerical groundwater simulation software PFLOTRAN with the energy infrastructure optimisation solver urbs. Therefore, in the next paragraphs, we describe the information required by each software, and how this information is used internally.

PFLOTRAN

The role of PFLOTRAN is to determine the subsurface temperature and pressure resulting from a specific set of GWHPs operating within the subsurface domain but does not have a boundary condition specific for modelling a GWHP. However, a GWHP can be modelled by a set of two separate source term boundary conditions, one for the injection well and one for the extraction well. The first boundary condition is the injection well, which specifies a time-series of injection temperatures $\boldsymbol{T}^{inj}$ $[°C]$ of a fluid injected into the subsurface, and a time-series of mass flow rates $\dot{\boldsymbol{m}}^{inj}$ $[\ell/s]$. The time series dataset specifies the temperature and mass flow rate at a specific time after the start of the simulation, e.g., if the simulation starts at day 0 and proceeds to day 365, then the temperature and mass flow rate can be specified in pre-set intervals (non-constant time intervals can be specified). An example is shown in Table 6.1.

The boundary condition for any time step within the PFLOTRAN simulation between these intervals are linearly interpolated by PFLOTRAN. The extraction well is modelled in the same manner, except that the negative mass flow of the injection well is defined for PFLOTRAN to extract the water out of the domain at the same mass flow rate. No temperature boundary condition is necessary.

urbs

The role of urbs in the simulation-optimisation coupling is to select the optimal set of GWHPs from a list of hypothetical GWHPs (potential locations where a GWHP could exist) in the city and their usage amounts (mass flow rate) based on the current state of the groundwater throughout

TABLE 6.1 Boundary conditions required to model the injection well of a GWHP in PFLOTRAN. The injection well temperature T^{inj} $[°C]$ and injection mass flow rate $\dot{m}^{inj}$ $[\ell/s]$ are required at various time steps [days].

Time step $[day]$	T^{inj} $[°C]$	$\dot{m}^{inj}$ $[\ell/s]$
0	13.4	0.028
10	13.98	0.0534
20	14.6	0.117
⋮	⋮	⋮
360	8.53	0.024
365	8.27	0.038

the domain. The selected set of GWHPs must not only satisfy the optimisation objective function, but also a set of external constraints[1]. To perform the optimisation and constraint checking, urbs requires:

- the groundwater temperature at all extraction wells, injection wells, and observation points,
- the groundwater pressure at all extraction wells, injection wells, and observation points.

Urbs is divided into two parts: *(i)* the main urbs script itself (referred to only as urbs) and *(ii)* the LP solver(optimiser). Urbs acts as an interpretation layer between the LP solver and the input configuration model, which stores the energy system infrastructure model and the energy demand information. The software Pyomo[2] is called from urbs to build the LP model (including the objective function and constraints) from the available information in the input configuration model. Therefore, any Pyomo compatible LP solver can be used in the optimisation process.

The information required from urbs for coupling are: *(i)* which GWHPs, selected from a set of hypothetical GWHP locations, are active in the city, *(ii)* what the mass flow rates for each of these GWHPs are and *(iii)* what the injection well temperatures are. The output from urbs optimiser (LP solver) is the energy-flux or energy rate $\dot{Q}$ at each GWHP in the model to meet the space heating and cooling demands. Therefore, the mass flow rates for each GWHP in the city can be calculated from Equation 6.1, which provides the information for the right column in Table 6.1,

$$\dot{m} = \frac{\dot{Q}}{c_p \Delta T}, \tag{6.1}$$

where $\dot{m}$ is the mass flow rate, c_p is the specific heat of water (assumed constant) and $\Delta T = T^{inj} - T^{ext} = 5$ is the constant temperature difference between the extraction and injection wells. The constant temperature difference is a physical setting of the GWHP itself, which always uses a temperature change of 5°C through the heat exchanger. The difficulty for modelling a

[1]These are constraints not used in the LP model itself, but within urbs to check that the GWHPs are operating within specified limits.

[2]`https://www.pyomo.org/`

GWHP in PFLOTRAN is that the temperature at the injection well depends on the temperature at the extraction well, which can be provided accurately only in a simulation including the respective GWHP. This requires an iterative process, performing a full groundwater simulation and adapting the injection well temperature to the extraction well temperature ±5°C in each iteration.

The urbs optimisation model requires the coefficient of performance (COP) for each GWHP, which is a function of the extraction well temperatures. The COP is the ratio of the space heating or cooling demand that can be met per energy (electrical) input unit. Therefore, a GWHP with a larger COP can meet the same demand while using less electricity, which is therefore cheaper than an alternative GWHP with a smaller COP. The cost minimisation objective would choose the cost-optimal solution to meet a demand. Therefore, the COP can directly influence the behaviour of the LP solution. The COP is determined based on the difference between the extraction well temperature $\boldsymbol{T}^{ext}$ and the load demand curve temperature $\boldsymbol{T}^{\ell dc}$, which is the time series temperature information specific to the respective heated or cooled building. Repeating the formulation from Equation 5.1, the COP is determined by

$$\boldsymbol{COP} = 9.97 - 0.2 \cdot \left(\boldsymbol{T}^{ext} - \boldsymbol{T}^{\ell dc}\right) + 0.0012 \cdot \left(\boldsymbol{T}^{ext} - \boldsymbol{T}^{\ell dc}\right)^2 .$$

In addition to the extraction well temperature to determine the COP, additional variables must be provided to urbs for the external constraint checking, which determines if the GWHPs are acting in the defined operating range. The external constraint checking procedure is performed within urbs and not in the LP solver itself. This means, that the LP solver suggests new GWHPs and their operating configuration without considering all external constraints, urbs checks the fulfilment of the external constraints and, accordingly, rejects or accepts the suggestion. The constraints are:

- injection well temperatures must be between 4°C and 20°C for all newly added GWHPs to the optimal solution (see hypothetical GWHPs below),
- pre-existing GWHPs that already exist must not be exposed to more than 1°C temperature change at the extraction well,
- pressure at the injection wells may not go above a predetermined value,
- pressure at the extraction wells may not drop below a predefined draw-down value.

During the optimisation process, it is necessary to keep track of the different types of GWHPs in the city, as this affects the constraint checking and LP optimisation. Therefore, we distinguish between five different types of GWHPs that are used throughout the optimisation.

- **Existing—original**: These are the already existing heat pumps that are currently in operation throughout the city. In the urbs optimisation model, they have a predefined usage amount that cannot change, and an already measured temperature difference between the extraction and injection wells. Each system may have multiple extraction and injection wells.
- **Existing**: This includes the existing—original, plus any other GWHP that is added to the

optimal solution throughout the coupled simulation-optimisation process. All GWHPs that satisfy the optimal solution and meet operational constraints are added here. Any new GWHP added to the list comes from the Selected list (see below for the description of this list).

- **Hypothetical**: The initial list of all potential locations where a GWHP can be added to the optimised model. Hypothetical locations are decided before the simulation begins based on physical constraints, difficulty in reaching a well site, houses or buildings occupying the space etc. They consist of only one extraction and one injection well each.
- **Selected**: This is the list of GWHPs that were added in the current iteration (see below). This list can only be populated from the hypothetical list. During our optimisation process, these heat pumps can be moved to the existing list or to the removed list.
- **Removed**: This is the list of GWHPs where the external constraints were not met, and which are permanently removed from consideration in all future iterations of the optimisation solver (by setting the maximum capacity to zero).

6.1.2 Coupling Scheme

The partitioned coupling scheme not only defines the flow of information between solvers but controls the operation of each solver. The coupling between urbs and PFLOTRAN consists of an outer iteration, where heat pumps are suggested by urbs and an adapted groundwater simulation with these new pumps is performed. Within a single outer iteration, multiple inner iterations are performed. Within the inner iterations, the GWHP infrastructure does not change, but only iterates over several PFLOTRAN simulations until the correct temperature difference between the extraction and the injection wells of all GWHPs is achieved. Both iterations are executed as a coupling between urbs and PFLOTRAN, where the outer iteration is mapped to what is interpreted as time steps of solvers in classical multi-physics simulations and the inner iterations are interpreted as the classical coupling iterations within each time step.

Outer Iterations

The aim of the urbs optimisation model is to find the optimal set of GWHPs within a region. Due to the external constraint checking that is performed and due to the black-box nature of the subsurface simulation, only one GWHP is added to the optimal solution in each urbs model in each outer iteration. This one-by-one addition of heat pumps allows urbs to correctly identify which GWHP has caused a potential violation of the constraints. Many outer iterations are required to add more GWHPs throughout the city, while maintaining a solution within the defined external constraints. The steps within a single outer iteration are defined as follows:

1. urbs receives the temperature values $\boldsymbol{T}^{ext}$ and $\boldsymbol{T}^{inj}$ and pressure values $\boldsymbol{P}^{ext}$ and $\boldsymbol{P}^{inj}$ at all extraction and injection wells from PFLOTRAN. The COP is determined for each GWHP (existing, selected, hypothetical, and removed).
2. The LP optimisation solver determines which hypothetical GWHP must be added to the

optimal solution, what the energy flux $\dot{\boldsymbol{Q}}$ and the mass flow rate $\dot{\boldsymbol{m}}$ is for the new GWHP. All existing GWHPs have a minimum and maximum capacity set to its own operating conditions such that they are always included in the LP solution.

3. The new GWHP is moved from the hypothetical to the selected list of GWHPs.
4. The injection well temperature for all GWHPs is determined by $\boldsymbol{T}^{inj} = \boldsymbol{T}^{ext} \pm 5$.
5. Multiple inner iterations are performed until the thermal groundwater simulation has converged to a stable solution.
6. The external constraints are checked for the newly added GWHP.
7. If the constraints are satisfied, the new GWHP is moved from the selected to the existing list, otherwise it is moved to the removed list and can never be considered for the optimal solution again (maximum capacity is set to zero).

Inner Iterations

For the urbs optimisation procedure to provide the correct operating conditions of the GWHPs, including all interaction between heat plumes of different GWHPs, urbs needs a converged temperature and pressure field of the subsurface domain. As explained above, PFLOTRAN needs to execute several simulations of the groundwater temperature field with incrementally adapted boundary conditions for the injection temperature at the injection wells of all GWHPs, until a converged solution with the prescribed temperature difference of 5°C between injection and extraction well is achieved. These iterations are our inner iterations. Although they only concern the groundwater simulation, they are technically implemented as an urbs-PFLOTRAN coupling via preCICE. It comprises the following steps:

1. urbs uses the extraction well temperatures from the previous iteration and suggests injection well temperatures via the formula $\boldsymbol{T}^{inj} = \boldsymbol{T}^{ext} \pm 5$°C.
2. The new $\boldsymbol{T}^{inj}$ values are provided to PFLOTRAN, and PFLOTRAN runs the subsurface simulation to provide an updated thermal and pressure field, in particular updated extraction well temperatures.
3. Urbs receives the updated temperature values at each extraction well.
4. Step 1 to 3 are repeated until the difference between successive $\boldsymbol{T}^{ext}$ values is below a defined threshold (convergence in preCICE implicit-coupling terms).

Overview of the complete coupling

In each PFLOTRAN solver run, the entire simulation from $t_{start} = 0$ to t_{end} is performed. Therefore, each coupling inner iteration involves a complete high-fidelity simulation. Likewise, when the LP solver is called in the first inner iteration of each outer iteration, the LP solver optimises the solution over the complete time period, from $t_{start} = 0$ to t_{end}. This contrasts with the simulations performed in Part I, where only one time step of the high-fidelity solver is called in each coupling iteration. A flow chart for the simulation solver control during the coupling runtime is shown in Figure 6.1, starting at point 1 "*Run PFLOTRAN*".

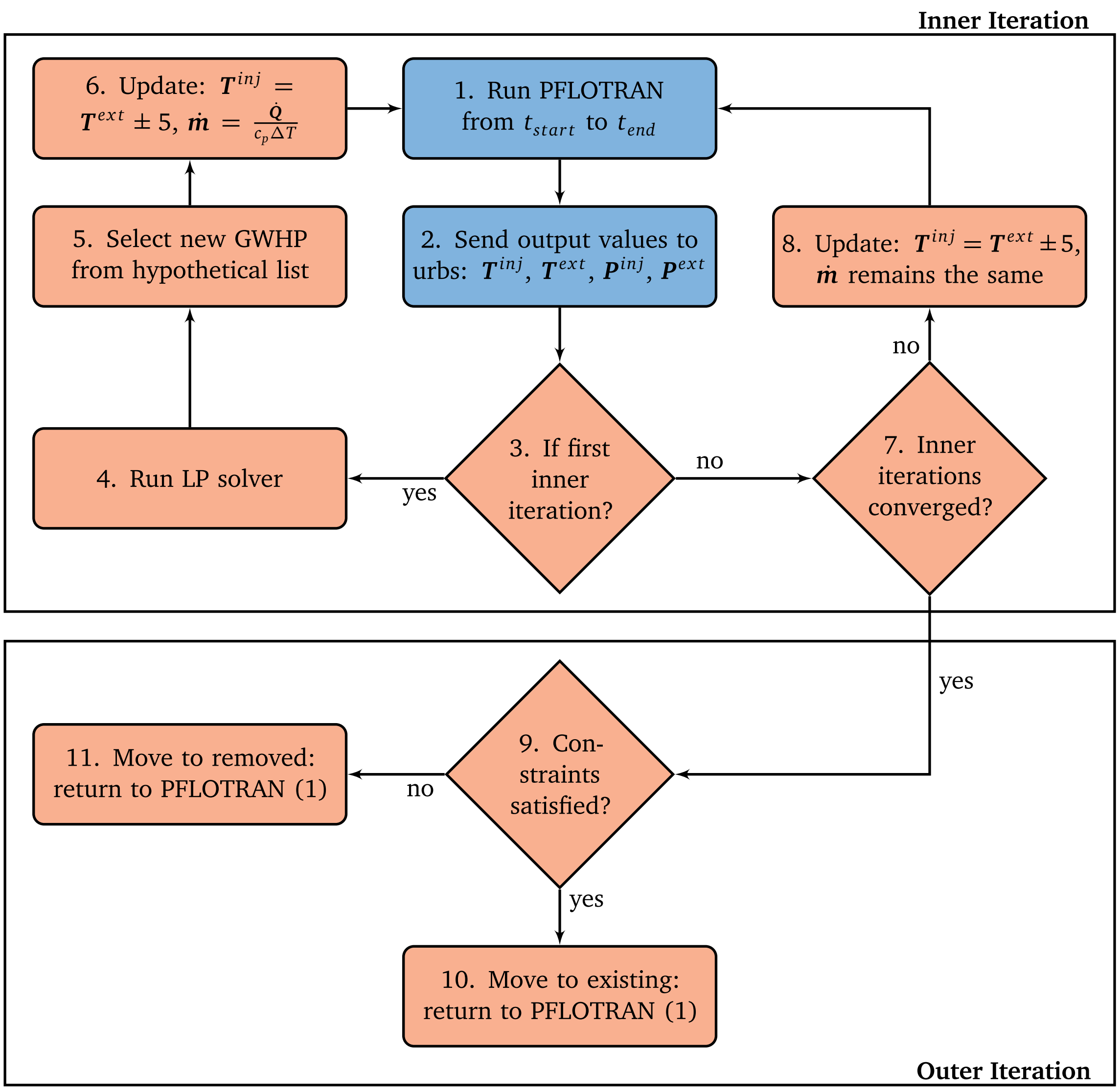

Inner Iteration
6. Update: $T^{inj} = T^{ext} \pm 5$, $\dot{m} = \frac{\dot{Q}}{c_p \Delta T}$
1. Run PFLOTRAN from t_{start} to t_{end}
5. Select new GWHP from hypothetical list
2. Send output values to urbs: T^{inj}, T^{ext}, P^{inj}, P^{ext}
8. Update: $T^{inj} = T^{ext} \pm 5$, $\dot{m}$ remains the same
no
4. Run LP solver
yes
3. If first inner iteration?
no
7. Inner iterations converged?
yes
11. Move to removed: return to PFLOTRAN (1)
no
9. Constraints satisfied?
yes
10. Move to existing: return to PFLOTRAN (1)
Outer Iteration

FIGURE 6.1 Flow diagram of inner-outer iteration procedure, starting from *Run PFLOTRAN*. In the first inner iteration, the LP solver is run to select a new GWHP, whereas only the injection temperatures T^{inj} are updated for all other inner iterations. Once the inner iterations have converged, the external constraints are checked to either keep or remove the selected GWHP. Steps in PFLOTRAN are indicated in blue, and steps in urbs are indicated in orange.

6.1.3 Parallelisation of the Coupled Simulation and Optimisation

Domain Decomposition

The coupled optimisation and simulation of the groundwater and energy system infrastructure over large areas and time spans, such as for the city of Munich over two or three years, is an extremely costly computational task. Thus, a domain decomposition approach to parallelise our framework to be able to run on powerful supercomputers is required. In particular, a smart method that can decompose the urbs problem into smaller domains is required. However, also during the construction and meshing of the PFLOTRAN domain it was observed that for the groundwater features to be resolved sufficiently (such as around GWHPs, culverts etc.), a single model of the entire city would be too large to run within a reasonable time frame. Therefore, the domain decomposition was also required to divide the single PFLOTRAN domain of the whole city of Munich into several smaller domains, each containing many smaller urbs regions.

Combined with the PFLOTRAN domain decomposition, multiple urbs regions must be generated for each PFLOTRAN sub-domain. As we will see in the following, if each PFLOTRAN sub-domain contained one urbs region, then each outer iteration would only add one GWHP to the entire PFLOTRAN sub-domain.

The domain decomposition is first performed for PFLOTRAN in order to obtain multiple PFLOTRAN sub-domain. The PFLOTRAN sub-domains are completely independent with no interactions considered. Thus, as the objective of the domain decomposition method was to limit the amount of GWHP interaction within each domain to minimise dependencies between the simulation and optimisations for different sub-domains. As the thermal plume of each GWHP tends to follow the velocity streamlines of the groundwater, a domain clustering method was developed utilising the velocity streamlines of the subsurface flow. This allows the clustering to define physically (almost) independent regions.

Standard analytical formulas, notably the linear advective heat transport model (LAHM) from [Kin87] are state-of-the-art for estimating the size of the thermal plume propagating from a GWHP. However, there is still considerable error for subsurface problems with heterogeneous subsurface parameter fields. Assuming a uniform background temperature, the temperature difference around a GWHP is approximated as

$$\Delta T(x,y,t) = \frac{Q \cdot \Delta T_{inj}}{4 \cdot n_e \cdot M \cdot v_a \cdot \sqrt{\pi \cdot \alpha_T}} \cdot exp\left(\frac{x-r}{2 \cdot \alpha_L}\right) \cdot \frac{1}{\sqrt{r}} \cdot erfc\left(\frac{r - v_a \cdot t/R}{2 \cdot \sqrt{v_a \cdot \alpha_L \cdot t/R}}\right) \tag{6.2}$$

where $\Delta T(x,y,t)$ is the time dependent temperature difference at coordinates x and y, ΔT_{inj} is the difference between the re-injection temperature and background temperature, v_a is the seepage velocity at x and y, r is the radial distance from the injection well, M is the aquifer thickness, and α_L and α_T are the longitudinal and tangential dispersivity values. The solution of this so-called LAHM model was overlayed onto the velocity streamlines, beginning at the GWHP site, and morphed to follow the streamline. The GWHP sites were added at thousands of locations throughout the city of Munich at locations viable for installation of a GWHP. A schematic of the subsurface velocity streamlines without thermal plumes and with the thermal plume prediction using the LAHM model is shown in Figure 6.2.

Based on this approximation, the areas are clustered into one sub-domain where there are significant overlaps of the thermal plumes. This limits the interaction between different PFLOTRAN models, improving the performance of the decomposed optimisation procedure. A k-means clustering procedure was used to define *k* sub-groups of GWHPs that have a significant number of overlapping thermal plumes. Must-links were created between two overlapping plumes, such that a single plume exists in at most two sub-domains.

The same procedure was performed to create multiple urbs regions within each PFLOTRAN sub-domain. All GWHP sites located within a PFLOTRAN sub-domain were used to create a set of overlapping plumes using the LAHM model, followed by a k-means clustering of the sites to create a set of urbs regions (same manner as the PFLOTRAN clustering above). Ultimately, the clustering procedure led to a set of sub-domains for PFLOTRAN and one set of urbs sub-domains per PFLOTRAN sub-domain. After the clustering procedure was performed for PFLOTRAN, the sub-domains were adjusted so that the almost vertical boundaries between domains follow a velocity streamline. This ensures that the boundary condition between two adjacent regions (left and right of each other) could be easily defined (with no velocity in/out of the boundary) and, at the same time, also minimises the number of thermal plumes that should propagate into an adjacent PFLOTRAN sub-domain. Within each PFLOTRAN sub-domain exist many smaller urbs models. The city of Munich divided into 31 different PFLOTRAN sub-domains is shown in Figure 6.3 (left) and the urbs sub-domains within a single PFLOTRAN sub-domain are shown in Figure 6.3 (right).

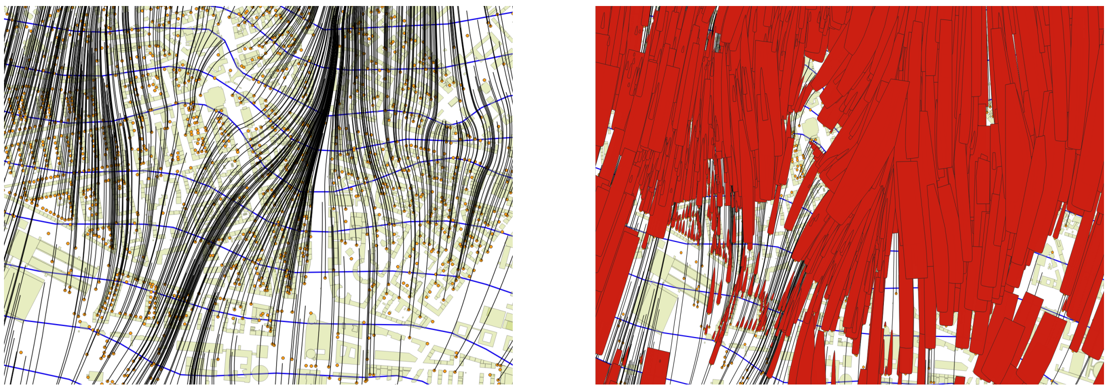
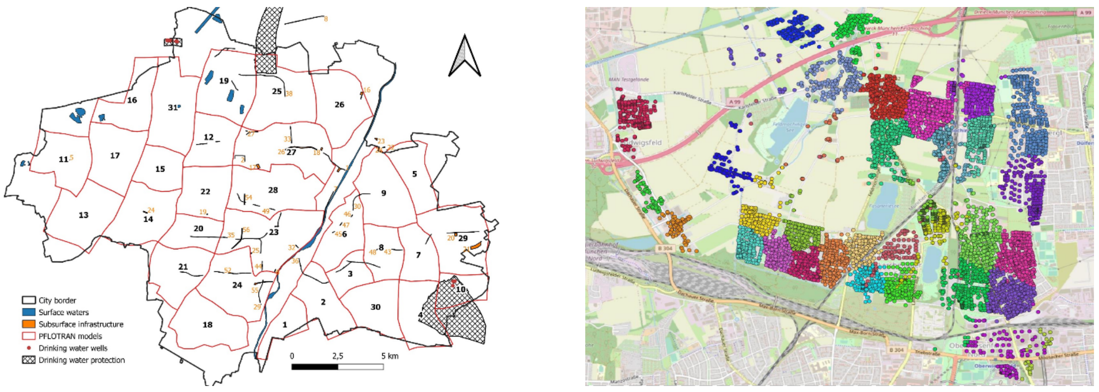

(A) Streamlines without plumes (B) Streamlines with plumes

FIGURE 6.2 Velocity streamlines without (left) and with (right) thermal plume approximation based on the LAHM model. The hydrostatic pressure isolines (blue lines) and the velocity streamlines (black) are depicted in both images. The velocity magnitude and subsurface properties are used to determine the plume extension (right) for each heat pump. The thermal plume prediction is then overlayed at the heat pump location, and morphed to follow the streamline direction. The red plume indicates a change of more than 1 °C from the LAHM formulation in Equation 6.2.



(A) PFLOTRAN sub-domains (B) urbs sub-domains

FIGURE 6.3 Domain decomposition of the PFLOTRAN and urbs models. The entire city was decomposed into 31 PFLOTRAN sub-domains (left). Each PFLOTRAN sub-domain was further decomposed into *k* urbs sub-domains, indicated by the different colours (right) for the optimisation with urbs.

Staggered Coupling Scheme

Due to the dependency on the flow simulation, not all constraints can be directly integrated in the optimisation in urbs. Thus, they need to be checked after tentatively adding GWHPs in the urbs optimiser. This induces the need to identify which GWHPs caused a violation of a constraint to remove the offending GWHPs. To ensure this, we developed a so-called staggered coupling scheme. In addition to the constraint checking issue, each urbs sub-domain is a completely independent optimisation problem that does not see any information from neighbouring regions. This lack of communication between the urbs regions is problematic. Consider two urbs regions with one region directly downstream of the other. The thermal plume that is created in PFLOTRAN by a GWHP operating in one urbs region may propagate downstream and negatively influence a GWHP operating in different urbs region. If a GWHP was added into both urbs regions within one outer iteration, and if the constraints were not satisfied in the downstream region, the downstream region would not know which GWHP has caused the problem. Thus, not all regions can be optimised in each outer iteration in our domain decomposition approach. Therefore, a staggered coupling scheme, which alternately switches urbs regions on and off for optimisation between outer iterations, was devised. The various urbs models are parallelised by using the Message Passing Interface (MPI) to help define which urbs regions need to run on our distributed computing systems in the next step. When decomposing the urbs models, each region has a globally unique region number, an MPI rank number, and the globally unique region number of its downstream neighbour, i.e., each region knows which other region it might affect, and which region it must query for a potential violation of constraints due to a GWHP added in its own region. As we alternate between two different sets of urbs regions in this approach, two regions (one from each set) are run on each computing rank.

The staggered coupling scheme is illustrated using an example of 8 GWHPs across 4 regions running on 2 ranks in Figure 6.4. Each region has two hypothetical GWHPs, where the extraction well (orange dot) and the injection well (blue dot) are connected via the black arrow between them which forms a single GWHP system. For each GWHP system, water is extracted at the orange dot point, run through the heat exchanger and reinjected at the blue dot site. The blue arrow indicates the direction of the subsurface flow at the GWHP (left to right for this example). Region 1 and region 2, called R_1 and R_2, are run on $Rank_1$ in an alternating fashion. Regions R_3 and R_4 are run on $Rank_2$ and alternated similarly.

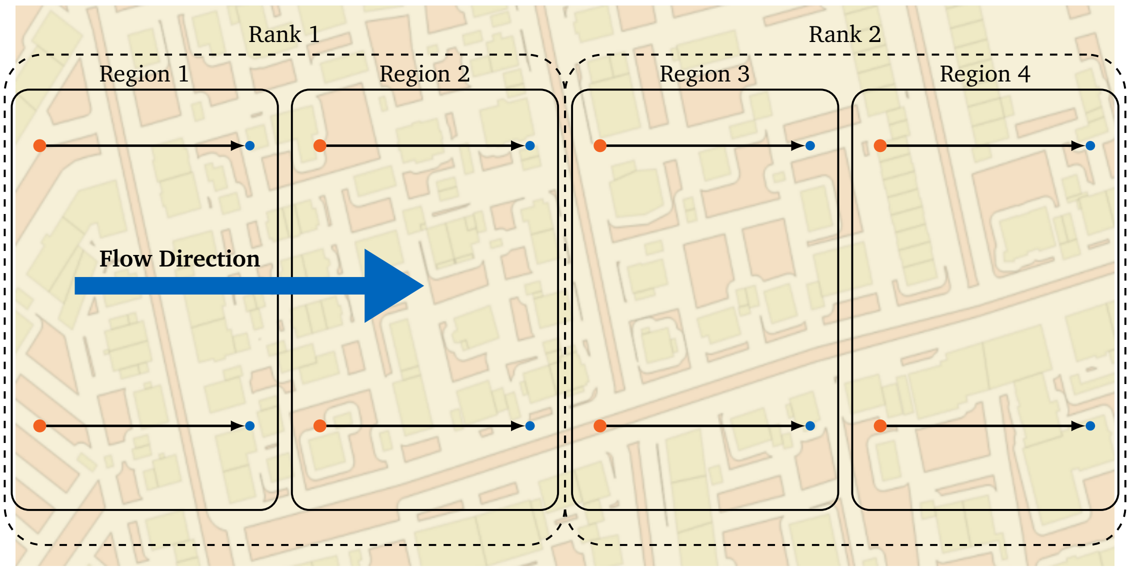


FIGURE 6.4 Initial configuration of our four-region model showing the inner-outer coupling procedure with constraint checking. Each region has two hypothetical GWHPs, and the thermal plume from a GWHP affects at most one downstream region. The extraction well (orange dot) and the injection well (blue dot) for a single GWHP system are connected by the black arrow between them. The GWHPs are placed onto a map of the city for perspective.

For each region, the global region number, the local region number (the order of running the regions on each rank), the rank number and the list of all impacted downstream regions are shown in Table 6.2. R_1 and R_2 are run in the same order as listed, both on $Rank_1$. R_1 affects the downstream region R_2 (right column), and R_2 affects the downstream region R_3. R_3 and R_4 are run on $Rank_2$, where R_3 affects R_4, and R_4 only affects itself (there are no more downstream regions). Even though our decomposition approach is presented for four regions placed neatly one after the other, the principle of the method generalises to any layout of regions and any number of downstream regions. However, the more "structured" the urbs regions are, and the smaller the number of downstream regions is, the better the optimisation results are.

The staggered coupling scheme operates as follows. At the start of the first inner iteration within the first outer iteration, the LP solvers in R_1 and R_3 (the first local region on each rank,

TABLE 6.2 Four-region model example: region number, computing rank, and impacted downstream regions for all urbs regions. The table is generated during the domain decomposition process and stored in a central database (Section 6.1.4).

Global Region Number	Local Region Number	Rank	Downstream Region
1	1	1	2
2	2	1	3
3	1	2	4
4	2	2	4

indicated by the green highlighting) select a GWHP from the available hypothetical list of GWHPs. As an example, the top GWHP in R_1 is selected and the bottom GWHP in R_3 is selected. After numerous inner iterations, without re-running the optimiser, a converged solution of the temperature and the pressure field is reached within PFLOTRAN. Figure 6.5 schematically shows the thermal plumes, indicated by the red ellipses originating from the injection wells (blue dots), and overlapping the extraction wells (orange dot) in R_2 and R_4. The active region on each rank is highlighted in green. The GWHPs in R_2 and R_4 are not active GWHPs, they are still in the hypothetical list. Therefore, the constraints are not considered at these locations. All constraints are satisfied in this case, and the GWHPs in R_1 and R_3 are moved to the existing list. This concludes one outer iteration.

At the start of the second outer iteration (Figure 6.6), the optimiser in R_2 and R_4 are run. The top GWHP in R_2 and the bottom GWHP in R_4 are influenced by thermal plumes from R_1 and R_3 and have a reduced COP[3]. Therefore, the optimiser selects the bottom GWHP in R_2 and the top GWHP in R_4. After a number of inner iterations, the external constraints are checked. The thermal plume from R_2 overlaps with the existing GWHP in R_3, as shown in Figure 6.6, and causes a large change in the extraction well temperature at R_3. As R_3 lies downstream of R_2, as specified in Table 6.2, the GWHP in R_2 violates the constraints and is moved into the removed list. The GWHP in R_4 does not impact any other GWHP and is moved into the existing list. This concludes the second outer iteration.

At the beginning of the third outer iteration (Figure 6.7), R_1 and R_3 are run again. The only available GWHP in R_1 is the bottom GWHP which is selected. Similarly, the top GWHP is selected in R_3, as shown in Figure 6.7. After the convergence of the inner iterations, the constraints are checked. The bottom GWHP in R_1 does not affect any other existing GWHP in R_1 or R_2 and is moved to the existing list. However, the top GWHP in R_3 impacts the existing GWHP in R_4. Therefore, the new GWHP in R_3 is moved to the removed list. This concludes the third iteration.

In the final outer iteration (Figure 6.8), all still available final available hypothetical GWHPs are impacted by upstream thermal plumes, which reduces their COP. Therefore, the optimiser will not select these GWHPs, and the final optimised result of the coupling procedure is shown in Figure 6.8.

[3]This, of course, depends on the constructed LP model, but we simplify the LP models for this example.

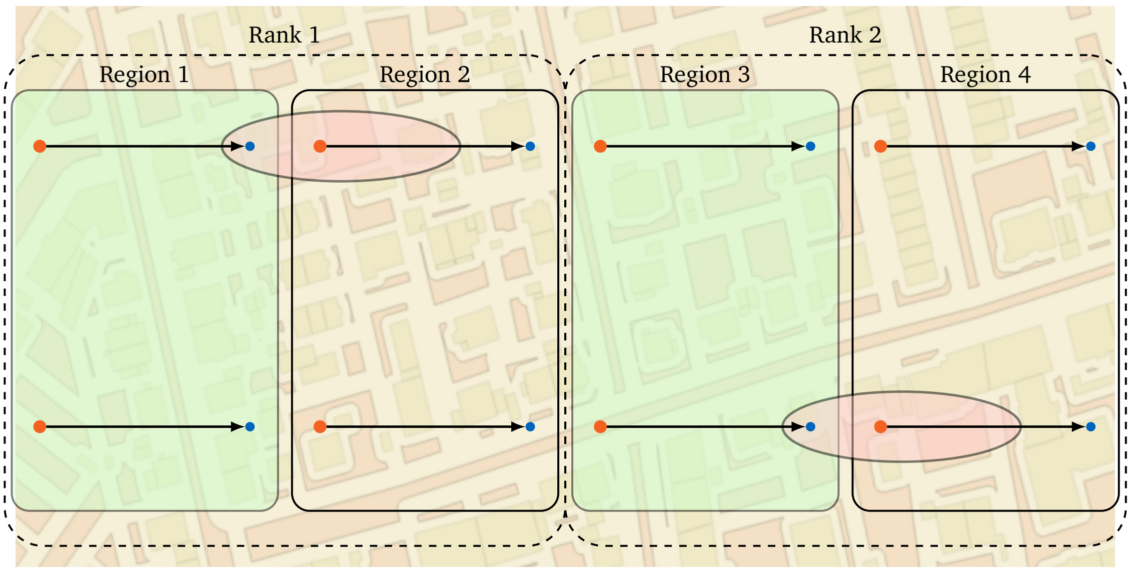


FIGURE 6.5 Result of the first optimisation step for our four-region model with two GWHPs selected in R_1 and R_3. The thermal plume from R_1 extends into R_2. Likewise, the thermal plume from R_3 extends into R_4. As no GWHP in R_2 or R_4 exists, all constraints are satisfied.

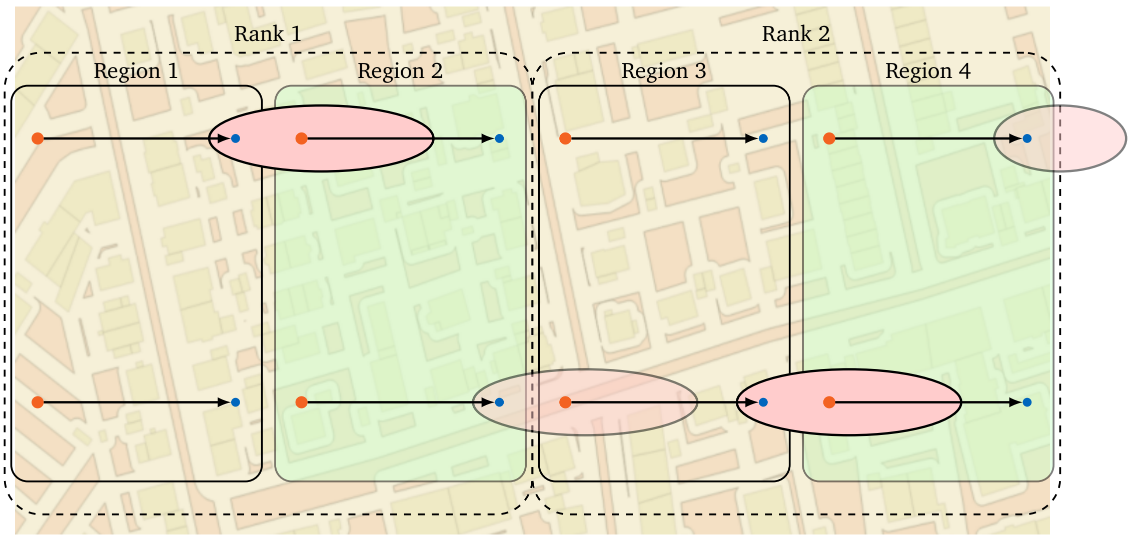


FIGURE 6.6 Result of the second optimisation step for our four-region model with two new GWHPs selected in R_2 and R_4 in the second outer iteration. The GWHP in R_2 interferes with the existing GWHP in R_3. Therefore, the GWHP in R_2 is removed. All constraints are satisfied in R_4.

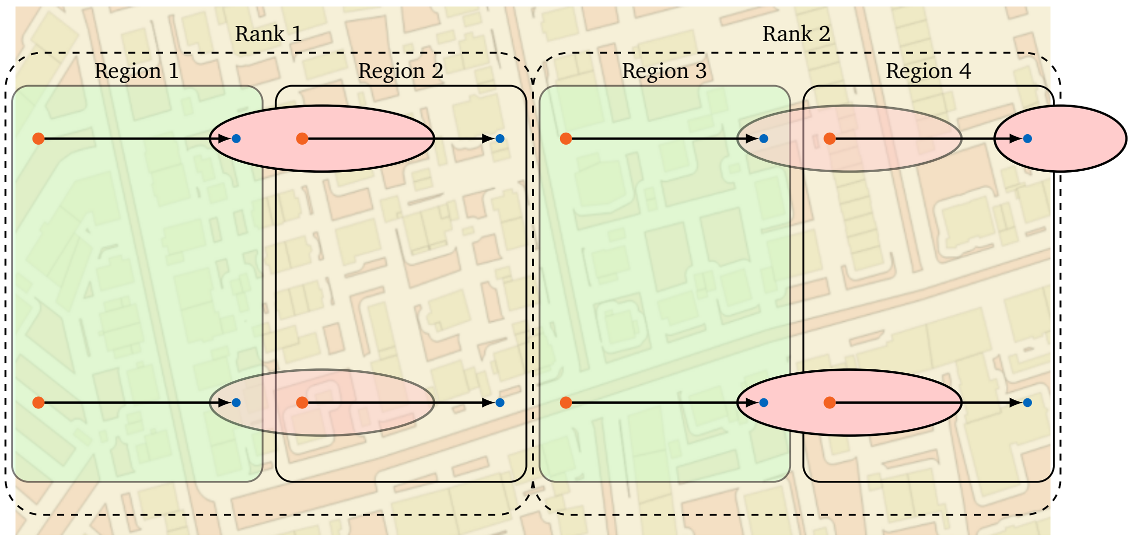


FIGURE 6.7 Result of the third optimisation step for our four-region model with two new GWHPs selected in R_1 and R_3 in the third outer iteration. The GWHP in R_1 does not interfere with any existing downstream GWHPS in R_2. However, the new GWHP in R_3 interferes with the already existing GWHP in R_4. Therefore, the GWHP in R_3 is removed.

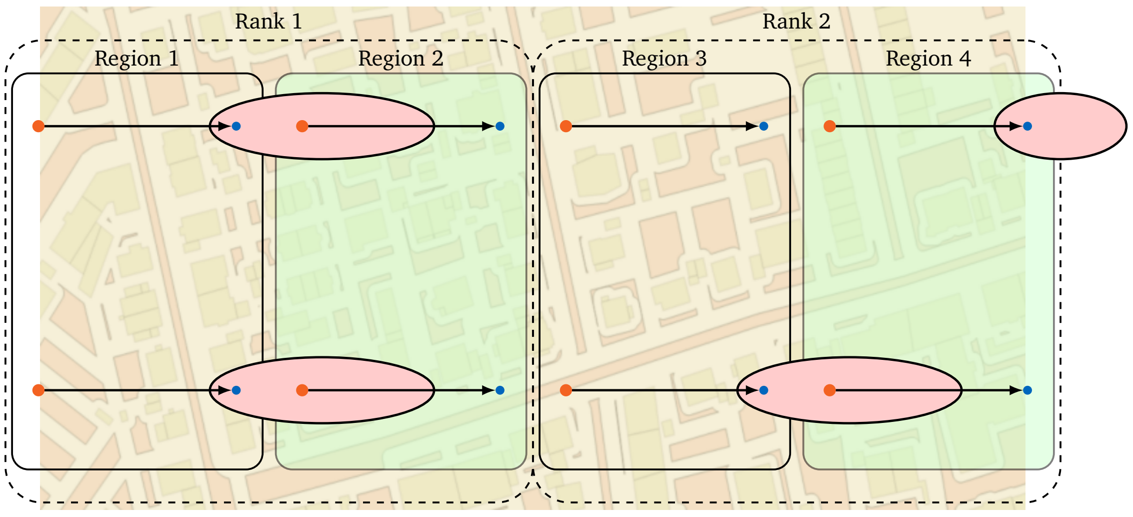


FIGURE 6.8 Final solution for the 8 GWHP, 4 region, 2 rank staggered coupling optimisation procedure. This is the known optimal solution for a setup where the COP is reduced when the extraction well is within the thermal plume of an upstream GWHP.

6.1.4 Parallel Data Mapping

Model Pre-Processing – Database Creation

There is a significant amount of pre-processing to prepare the PFLOTRAN-urbs simulation-optimisation coupling. In this section, the central database that stores all the coupling data and is used to create the necessary coupling configuration files, is described.

Creating the PFLOTRAN and urbs models and performing the coupled simulations requires detailed information of the locations of GWHPs, operating constraints of existing-original GWHPs and estimated energy demands. All this information is stored in a centralised database, that allows for the data associated with each well for each GWHP to be available for a pre-processing step to generate the input models. In this section, we describe the database format and how this is used to generate all input files to perform a coupled simulation.

The information available for each well of every GWHP is stored in a `well-summary` database, with an example output shown in Table 6.3. For each well (first column), information such as the x and y coordinates of the GWHP (where in the city the GWHP is), the $\texttt{Well}_{ID}$, $\texttt{System}_{ID}$, $\texttt{Cell}_{ID}$, `wellType`, $\texttt{Plot}_{ID}$ and $\texttt{P}_{max}$ are saved. The $\texttt{Well}_{ID}$ is a reference ID that is unique for every well drilled in the city, whereas the $\texttt{System}_{ID}$ is unique for each GWHP only (there can be multiple wells per system). The $\texttt{Plot}_{ID}$ is a reference within the urbs models that is unique for each GWHP system. The $\texttt{Cell}_{ID}$ is available after the meshing process in PFLOTRAN and is the globally unique cell ID within the finite volume PFLOTRAN mesh (Section 5.3.2) where the GWHP boundary condition is applied, and where the output observation is extracted from. The output observation is performed at an observation point, which is a point in space within the PFLOTRAN domain where the temperature and pressure value must be output by PFLOTRAN. Therefore, it is possible to obtain the unique $\texttt{Cell}_{ID}$ for each GWHP in the PFLOTRAN domain and in each urbs model once the domain meshing is completed.

TABLE 6.3 Database information for all injection and extraction wells, for all GWHPS.

Well Number	x	y	Well_{ID}	System_{ID}	Cell_{ID}	wellType	Plot_{ID}
0	68.96	230.27	F3F2FF	E1	1766971	E	F3F2FF
1	185.16	248.37	F3F327	E2	2412407	I	F3F327
2	231.16	139.27	9004534	9003242	451464	E	F43857
3	248.46	147.07	71863	59407	3715505	I	F41795
⋮	⋮	⋮	⋮	⋮	⋮		

Additional data that can be obtained from the database are the time-series constraint information for each well for: *(i)* the minimum pressure drawdown (which is the minimum allowable reduction in the water table height which is approximated by the groundwater pressure), *(ii)* the temperature difference between the extraction and injection wells for the existing-original

GWHPs, and *(iii)* the mass flow rate for the existing-original GWHPs in the city. An example output[4] of the mass flow rate for the existing-original GWHPs is shown in Table 6.4, with the Well$_{ID}$ and associated Cell$_{ID}$ for each well (columns) at various time steps (rows).

TABLE 6.4 Time series constraint information for the mass flow rate at Existing-original GWHPs. The constraints are specified using the unique Well$_{ID}$ and Cell$_{ID}$ for each Existing-original GWHP. The left column contains the time steps for which the constraint information is provided. Any constraint value between these time steps is linearly interpolated.

Well$_{ID}$	71804	69090	69091	69149
Cell$_{ID}$	42515	30213	23681	9253
time [day]				
0	-0.436	-0.066	-0.3018	-0.138
15	-0.459	-0.071	-0.316	-0.142
⋮	⋮	⋮	⋮	⋮
354.05	-0.244	-0.046	-0.104	-0.041
365	-0.107	-0.016	-0.092	-0.035

The database provides all of the information that can be used to generate the urbs and PFLOTRAN models to enable data mapping and additional constraint checking within each urbs model. This also allows for the PFLOTRAN and urbs model generation process to be automated, simplifying the model generation process and reducing model errors.

Data Mapping Method

The data mapping strategy between urbs and PFLOTRAN is critical to ensure that the coupling data for each GWHP at the correct time step are exchanged. As both urbs and PFLOTRAN require the coupling information at each time step for each location at the start of the simulation, the regular method used in preCICE, which is generating the coupling mesh using the vertex coordinates is no longer suitable, as the graph network of the urbs optimisation model does not have any inherent coordinates in space. Additionally, within the current mapping capabilities of preCICE, the exchanged information is applied to a vertex with a point in space that has no sense of time, as information is typically exchanged within a time step, whereas we exchange time series data at each location. Therefore, a new coupling interface is created using a globally unique identifier for each GWHP as the first spatial coordinate, and the time step of when the information was generated/must be applied as the second spatial coordinate.

In order to obtain the first spatial coordinate for the coupling interface, a unique identifier for each GWHP in both PFLOTRAN and urbs is required. The `Cell`$_{ID}$ is a globally unique number in the PFLOTRAN mesh, defining a specific finite volume element in the mesh. The `Cell`$_{ID}$ information is available in PFLOTRAN and can be provided to urbs (through knowledge in the central database) and be used to build the coupling interface. As this value is unique in both

[4]Actual data for the REG-30 model explained in Section 7.2

PFLOTRAN and urbs for each extraction and injection well and is not repeated across MPI ranks on distributed systems, this can technically be considered a point in a one-dimensional (1D) space. The vertices on the coupling interface can be generated as a combination of the $\texttt{Cell}_{ID}$ value and the time when the solvers exchange information , forming a 2-dimensional vertex with coordinates [$\texttt{Cell}_{ID}$, `time`].

The consistent nearest-neighbor mapping (Section 2.3.1) can be used to ensure that information at vertex [$\texttt{Cell}_{ID}$, $\texttt{time}_1$] in PFLOTRAN is copied to the vertex [$\texttt{Cell}_{ID}$, $\texttt{time}_1$] in urbs. Therefore, a mesh can be specified in preCICE as a list: `mesh` = [[$\texttt{Cell}_{ID_1}$, $\texttt{time}_1$], [$\texttt{Cell}_{ID_2}$, $\texttt{time}_2$],...,[$\texttt{Cell}_{ID_1}$, $\texttt{time}_{end}$, [$\texttt{Cell}_{ID_2}$, $\texttt{time}_1$], ..., [$\texttt{Cell}_{ID_{end}}$, $\texttt{time}_{end}$]]. The existing nearest-neighbor mapping procedure in preCICE already handles parallel data communication, simplifying the PFLOTRAN-urbs data exchange. The algorithm for finding the $\texttt{Cell}_{ID}$ for each GWHP within each solver is described below.

PFLOTRAN

Before the coupled interface information can be supplied to PFLOTRAN, the mesh is generated from the geometry and stored in a central database. The database stores the $\texttt{Cell}_{ID}$ for every injection and extraction well for each GWHP. PFLOTRAN automatically decomposes the meshes domain for parallelisation, i.e., the domain is divided into smaller regions that each run on one rank. Before running the simulation, it is not known how PFLOTRAN will decompose the domain, or on which computing rank a specific GWHP will be.

When applying the boundary conditions to the GWHPs in the PFLOTRAN model, i.e., when reading the temperature or the mass flow rate time series from the input file `pflotran.in`, the boundary conditions are read and applied to the wells for each GWHP in the order as they appear in `pflotran.in`. This is performed on each rank. When the `pflotran.in` file is created, the order of GWHPs with which PFLOTRAN will loop through when applying the boundary conditions is known. The list of sink/source boundary conditions (one associated with each well) is written to the input file in the order of hypothetical injection, existing-original injection, hypothetical extraction, existing-original extraction. As each boundary condition is applied at a PFLOTRAN "region", which specifies the $\texttt{Cell}_{ID}$ that the boundary condition must be applied to, the order of the $\texttt{Cell}_{ID}$ to build the `read_mesh` can be determined before the simulation is started, as it was already written to the input file `pflotran.in`. The `read_mesh` is the list of vertices that PFLOTRAN will query from preCICE to obtain information from urbs, which contains the new boundary condition information for PFLOTRAN. Therefore, the `read_mesh` vertices can be created by looping over all $\texttt{Cell}_{ID}$ and time steps according to Algorithm 6.1. This is only required for the injection wells as extraction wells are implicitly defined by their injection wells. At the start of each inner iteration, the new boundary condition data are read from preCICE by simply looping sequentially through the `read_mesh` and overwrites by the original boundary condition data. In a parallel setting, this is only performed on the main rank and, subsequently, new data from preCICE is globally distributed by a single `MPI.Scatter` command. This provides PFLOTRAN with the correct read information when both solvers are running in parallel.

```
if myRank == main then
  Load Cell$_{ID}$ list in order of boundary conditions in pflotran.in
  read_mesh = []                    ⇝ Create empty list of mesh to read preCICE data
  for id in Cell$_{ID}$ do
    for $j = 0, \dots, N_{TS}^{coupling}$ do       ⇝ for each time step value for coupling
      location = [$id$, $j$]
      read_mesh.append(location)                ⇝ append mesh by 2D vertex
Therefore:  read_mesh = [[$id_1$, $t_1$], [$id_1$, $t_2$], ..., [$id_1$, $t_{end}$], [$id_2$, $t_1$], ...]
```

ALGORITHM 6.1 Algorithm for building the `read_mesh` in PFLOTRAN.

The temperature and pressure values required by urbs are obtained from PFLOTRAN by creating observation points within PFLOTRAN. As defined before, an observation point is a point in space within the PFLOTRAN domain where the temperature and pressure value must be output by PFLOTRAN. The observation point location is defined by a Cell$_{ID}$, i.e., the temperature and pressure are output at all finite volume cells listed in the "Observation" section of the `pflotran.in` file. Due to the domain decomposition parallelisation, it is unknown which rank each output observation will exist on and needs to be determined during the simulation runtime. When PFLOTRAN writes the observation value for each well of every GWHP, the Cell$_{ID}$ is output alongside the observation value. Therefore, during the simulation runtime, the Cell$_{ID}$ value for each observation on each rank is obtained. In the first inner iteration, when the observation values are written for the first time, the `write_mesh` is created by writing the Cell$_{ID}$ to preCICE. This mesh is used to write the observation values at all injection and extraction wells to preCICE. Conversely to the `read_mesh`, the rank where the observation point Cell$_{ID}$ is only known locally after the simulation has begun, and therefore, only a portion of the entire coupling mesh is built on each rank. This is automatically accounted for in the data mapping function in preCICE. The steps in creating the `write_mesh` is shown in Algorithm 6.2.

```
write_mesh = []                     ⇝ Create empty list of mesh to write preCICE data
for obs in Output Observations do                ⇝ Observation on local rank only
  id = Cell$_{ID}$          ⇝ PFLOTRAN has the cell$_{id}$ when writing output observation
  for $j = 0, \dots, N_{TS}^{coupling}$ do         ⇝ for each time step value for coupling
    location = [$id$, $j$]
    write_mesh.append(location)                 ⇝ append mesh by 2D vertex
```

ALGORITHM 6.2 Algorithm for building the `write_mesh` in PFLOTRAN.

urbs

Each urbs region needs to create a `read_mesh` and a `write_mesh` that allows urbs to read data at all injection and extraction wells from preCICE and write data for all injection wells to preCICE.

Only the injection well temperature and mass flow rate are provided from urbs and the application of the extraction well mass flow rate is handled in PFLOTRAN. As described in the section above, the $\texttt{Cell}_{ID}$ value for each well and GWHP is required. After the PFLOTRAN domain is meshed and the database information is generated, the $\texttt{Cell}_{ID}$ and `welltype` (whether the well is an injection or extraction well) is known for each well belonging to every GWHP, along with the corresponding $\texttt{Plot}_{ID}$ for each GWHP system. The $\texttt{Plot}_{ID}$ is an identifier for each GWHP system in the urbs models.

Once each urbs model is loaded during the urbs initialisation phase, the $\texttt{Plot}_{ID}$ is available for each GWHP system. The $\texttt{Plot}_{ID}$ is not globally unique, as multiple wells have the same $\texttt{Plot}_{ID}$ if they belong to the same GWHP system. By looping through each GWHP in the urbs models, the $\texttt{Plot}_{ID}$ for each GWHP can be found and the database is searched for all wells with the corresponding $\texttt{Plot}_{ID}$, returning all $\texttt{Cell}_{ID}$ and `welltype` information. The `read_mesh` and `write_mesh` can be written in the order that the GWHPs exist in each urbs model and the order that the $\texttt{Cell}_{ID}$ values are returned to urbs. The data mapping procedure for urbs is shown in Algorithm 6.3.

```
write_mesh = [], read_mesh = []
for hp in GWHPs do                                  ⇝ Loop through each GWHP
  plot-id = Plot_ID[hp]
  for all Cell_ID with plot-id do
    id = Cell_ID
    if wellType == E then                           ⇝ If an extraction well
      for $j = 0, \dots, N_{TS}^{coupling}$ do      ⇝ Number of time steps for coupling
        read_mesh = [$id$, $j$]
    else
      for $j = 0, \dots, N_{TS}^{coupling}$ do
        read_mesh = [$id$, $j$]
        write_mesh = [$id$, $j$]
```

ALGORITHM 6.3 Algorithm for building the `read_mesh` and `write_mesh` in urbs.

6.2 Coupling Software

The staggered coupling scheme described above requires numerical and technical interaction between the individual software components. In this work, we use preCICE to provide all data exchange and communication between PFLOTRAN and urbs. The Application Programming Interface[5] (API) of preCICE allows for the easy integration of the preCICE function calls into a simulation software, provided that the source code is available. This section describes the implementation details of adding the preCICE API function calls to PFLOTRAN and urbs.

[5] https://precice.org/couple-your-code-api.html

6.2.1 Adapter Steps

Before discussing how the preCICE adapters are implemented in PFLOTRAN and urbs, the general procedure for coupling a simulation software with preCICE is discussed. The algorithm of a typical simulation software, including both urbs and PFLOTRAN, is generally split into three sections: *(i)* initialisation, *(ii)* simulation solver and *(iii)* finalisation. The initialisation step usually involves reading of input files, performing the domain decomposition, initialising memory and MPI channels and any other once off computation at the start of the simulation. The simulation solver step involves solving the expensive numerical simulation, such as the solver computing the solution from t_{start} to t_{end} in the case of PFLOTRAN. The finalisation step usually stores the results, frees up memory and closes any MPI communication channels.

The steps within a typical simulation solver, modified with preCICE API functions (blue) are shown in Algorithm 6.4. The solver is started, and the initialisation procedure is performed (line 1 and 2). Next, a solver interface is created, which lets preCICE know that the solver is running on rank "myRank" (line 3). The coordinates for each vertex on the coupling mesh that lie on rank "myRank" are sent to preCICE (line 4 and 5), which handles all the data mapping (see Section 6.1.4). The coupling initialisation steps conclude with the `precice.initialise()` call, which informs preCICE that the adapter is ready to begin the coupled simulation. Within each iteration of the time-dependent solver, coupling data are read from preCICE (line 8), and the boundary conditions of the initial problem are modified with the new data (line 9). The regular simulation `runSolver()` is called for one time step for typical multi-physics applications in Part I of this thesis, or from t_{start} to t_{end} in the case of PFLOTRAN in the current coupling problem. After the `runSolver()` is finished, the output data is written to preCICE and `precice.advance()` is called, which hands the control of the solver to preCICE. The solver then waits until a signal is returned from preCICE before continuing.

When `precice.advance()` returns a signal to continue, the convergence of the solution is checked. If the solution has converged, the solver moves on to the next outer iteration and the respective preCICE counter is incremented. When all outer iterations are complete, or no more GWHPs are added to the optimal solution, the simulation finalises (line 18 and 19). This is a general formulation of the coupling procedure with preCICE. However, the implementation needed for PFLOTRAN and urbs entails more complexities than this formulation and is explained below.

6.2.2 PFLOTRAN Adapter

The PFLOTRAN adapter was developed inside the source code of PFLOTRAN. A new executable was generated (`pflotran-precice.o`) that provides the coupled functionality with preCICE, while leaving the original executable untouched (`pflotran.o`). The preCICE API calls are implemented in a new file called `Adapter.F90`, resulting in minimal changes inside of PFLOTRAN

```
turnOnSolver();
initialiseSolver();
precice::SolverInterface precice("SolverName","precice-config.xml", myRank,
size);
double coordinates = new double[NumberOfVertices];  ⇝ coordinates of coupling vertices
precice.setMeshVertices(meshID, coordinates);       ⇝ Creates coupling mesh
dt = precice.initialise();
while t < $t_{end}$ do                              ⇝ Begin time loop
  precice.readData(inputDataID, vertexSize, vertexIDs, Inputs);   ⇝ Reads data from preCICE
  apply Boundary Conditions:  Inputs
  runSolver();
  Compute output data:  Outputs
  precice.writeData(outputDataID, vertexSize, vertexIDs, Outputs);   ⇝ Writes data to preCICE
  precice.advance();   ⇝ Gives control back to preCICE and waits for a command to continue
  if inner iterations converged then
    t += dt
precice.finalise();
TurnOffsolver();
```

ALGORITHM 6.4 General coupling adapter with preCICE API functions for a time-dependent simulation solver.

source code itself. Therefore, the regular PFLOTRAN solver can be used alongside our new solver that is coupled to preCICE. The pseudo-code describing the steps within the PFLOTRAN adapter is shown in Algorithm 6.5.

The preCICE-PFLOTRAN adapter begins by initialising the solver and reading all required input information. During the `initialise solver` step, the vertex information to create the coupling `read_mesh` is created. The domain is decomposed into smaller sub-domains that run on each compute rank, which are handled internally in PFLOTRAN. The user has no knowledge beforehand as to how the domain is decomposed.

The coupling sequence begins with the first inner iteration within the first outer iteration. In this case, no coupling data are available yet from preCICE, and the initial boundary conditions from the input file are used. However, for all other coupling iterations, the boundary condition information is read from preCICE: both the time series well temperatures $\boldsymbol{T}^{inj}_{(:,k)}$ and injection well mass flow rates $\dot{\boldsymbol{m}}^{inj}_{(:,k)}$ are only provided at the injection well locations. By limiting the read data to the injection wells only, the size of the surface coupling mesh is reduced (as $\dot{\boldsymbol{m}}^{ext}_{(:,k)} = -\dot{\boldsymbol{m}}^{inj}_{(:,k)}$).

After applying the boundary condition information, the numerical solver is started, and PFLOTRAN runs from time t_{start} to t_{end}. When the current time step t^{j+1} within the solver is at an output observation time step (a point in time where the temperature and the pressure are output at specific locations within the PFLOTRAN domain, known as observation points), the results for the injection well temperature $\boldsymbol{T}^{inj}_{(:,k)}$, the extraction well temperature $\boldsymbol{T}^{ext}_{(:,k)}$, the injection well pressure $\boldsymbol{P}^{inj}_{(:,k)}$ and the extraction well pressure $\boldsymbol{P}^{ext}_{(:,k)}$ for each GWHP k are stored. Once the solver has finished, the stored results are written to preCICE. This comprises the temperature and pressure at each location (injection and extraction wells) for all output time steps. Contrary to the urbs adapter, the PFLOTRAN adapter does not need to explicitly consider the beginning or end of the outer iterations as PFLOTRAN functions in the same manner in every iteration after the first inner iteration. Once all outer iterations are complete, PFLOTRAN is terminated, and the simulation-optimisation coupling is complete.

```

Begin PFLOTRAN
Initialise solver                                  ⇝ Read input file and decompose domain
precice::SolverInterface(PFLOTRAN), precice.setMeshVertices(),
precice.initialise()
for $i = 0, 1, 2, ...$ do                                ⇝ Start Outer Coupling Iterations
  if $i == 0$ then                                        ⇝ If first coupling iteration
    Read coupling data from pflotran.in for all $N_{gwhp}$ heat pumps: $\boldsymbol{T}^{inj}_{(:,k)}$,
    $\dot{\boldsymbol{m}}^{inj}_{(:,k)}$, $\dot{\boldsymbol{m}}^{ext}_{(:,k)}$, $\forall k \in N_{gwhp}$
  else
    Read coupling data from preCICE for all $N_{gwhp}$ heat pumps: $\boldsymbol{T}^{inj}_{(:,k)}$, $\dot{\boldsymbol{m}}^{inj}_{(:,k)}$,
    $\forall k \in N_{GWHP}$
    $\dot{\boldsymbol{m}}^{ext}_{(:,k)} = -\dot{\boldsymbol{m}}^{inj}_{(:,k)}$
    Overwrite boundary conditions with new data
  Begin time-dependent numerical simulation
  $m = 0$                                      ⇝ m tracks when output data must be stored
  for $j = 0, \ldots, t_{end}$ do
    Solve groundwater flow for a time $t^{j+1} = t^j + \Delta t_{sim}$
    if $t^{j+1}$ % $\Delta t_{coup\ell ing} == 0$ then        ⇝ Output observation at all observation points
      for $k = 1, \ldots, N_{gwhp}$ do
        Save output data
        $T^{inj}_{(m,k)}$ = Temperature$^{inj}_{k}$                 ⇝ Injection well temperature for GWHP k
        $T^{ext}_{(m,k)}$ = Temperature$^{ext}_{k}$
        $P^{inj}_{(m,k)}$ = Pressure$^{inj}_{k}$; $P^{ext}_{(m,k)}$ = Pressure$^{ext}_{k}$
        $m += 1$
  Write Coupling Data to preCICE: $\boldsymbol{T}^{inj}_{(:,k)}$, $\boldsymbol{T}^{ext}_{(:,k)}$, $\boldsymbol{P}^{inj}_{(:,k)}$, $\boldsymbol{P}^{ext}_{(:,k)}$
  Call preCICE Advance                           ⇝ preCICE controls information flow from here
  if If coupling is complete then
    break
```

ALGORITHM 6.5 Pseudo-code for the PFLOTRAN adapter for both inner and outer iterations coupled to preCICE. The variables are: the total number of GWHPs N_{gwhp}, the solver time step Δt_{sim}, the coupling time step for preCICE $\Delta t_{coup\ell ing}$. The subscript $(:, i)$ indicates the values for all coupling time steps for GWHP i.

6.2.3 urbs Adapter

The urbs adapter required a more intrusive approach to enable coupling with preCICE. Urbs is a python script (`urbs.py`) that reads in an urbs model, creates a Pyomo model that can interface with various LP solvers (such as the free software GLPK[6] or the commercial software Gurobi[7]) to solve the optimisation problem. The `urbs.py` solver was directly modified to include the preCICE API function calls, thereby creating a new solver `urbs_precice.py`. Therefore, the preCICE adapter is a new urbs solver, and not merely and addition to the original urbs solver. The preCICE API function calls provide the information to control when the LP solver is run and when the external constraint checking is performed.

The pseudo-code describing the urbs solver is divided into two parts: Algorithm 6.6 describes the inner iteration process and Algorithm 6.7 describes the outer iteration and external constraint checking process. When urbs is started, additional information for constraint checking is required: *(i)* the maximum groundwater pressure allowed at every hypothetical GWHP injection well $\boldsymbol{P}^{inj,nat}_{(:,k)}$, *(ii)* the minimum allowable pressure draw-down $\boldsymbol{P}^{drawdown}_{(:,k)}$ at each hypothetical GWHP extraction well, *(iii)* the extraction well mass flow rate $\dot{\boldsymbol{m}}^{ext,eo}_{(:,k)}$ at every existing-original GWHP, *(iv)* the temperature difference $\boldsymbol{T}^{diff}_{(:,k)}$ between the extraction and injection well for each existing-original GWHP, *(v)* the load demand curve temperature $\boldsymbol{T}^{\ell dc}_{(:,k)}$ for every existing-original and hypothetical GWHP, *(vi)* `globalRegionNumber`, which provides the globally unique urbs region number for each urbs region running locally on each rank, and *(vii)* `globalDownStreamRegionNumber`, which provides the globally unique number of the urbs region number that lies downstream.

When urbs is started on each compute rank, each rank loads the urbs models that are assigned to it in the `regionLocator` file, which stores the information in Table 6.2. Each urbs model is initialised, and each GWHP and its unique Plot_{ID} (Table 6.3) for each model are stored. The $\texttt{Plot}_{ID}$ value for each GWHP is used to obtain the $\texttt{Cell}_{ID}$ from a central database to define the coupling interface for each model.

In the first inner iteration of the first outer iteration, urbs does not receive any input from preCICE as both solvers run in parallel (using the `parallel-implicit` coupling scheme in preCICE). In the first inner iteration, the extraction well temperatures $\boldsymbol{T}^{ext}_{(:,k)}$ are used to update the COP values (line 10). The optimiser is run, and the energy flux $\dot{\boldsymbol{Q}}$ for each GWHP is obtained from the solution to determine the mass flux (line 12). The optimiser may have selected more than one hypothetical GWHP for the optimised solution, which would not work with the designed staggered coupling scheme. Therefore, only the hypothetical GWHP with the largest energy flux is added to the selected list (line 14 and 15). The boundary conditions for the injection temperature $\boldsymbol{T}^{inj}_{(:,k)}$ are updated according to three different operating conditions: existing-original, existing or selected GWHP with positive $\dot{\boldsymbol{Q}}$, or either an existing or selected GWHP with negative $\dot{\boldsymbol{Q}}$. Finally, the data at the injection wells are written to preCICE, and `precice.advance()` is called. If the

[6]`https://www.gnu.org/software/glpk/`
[7]`https://www.gurobi.com/`

```
Begin urbs
Initialise solver                                    ⇝ Read input file and decompose domain
precice::SolverInterface(urbs), precice.setMeshVertices(),
precice.initialise()
Load constraint checking data from input files: $P^{inj,nat}_{(:,k)}$, $P^{draw-down}_{(:,k)}$, $\dot{m}^{ext,eo}_{(:,k)}$, $T^{diff}_{(:,k)}$.
for $i = 0,1,2,...$ do                                             ⇝ Begin Outer Iterations
  for $j = 0,1,2,...$ do                                           ⇝ Begin Inner Iterations
    Read Coupling Data from preCICE: $T^{inj}_{(:,k)}$, $T^{ext}_{(:,k)}$, $P^{inj}_{(:,k)}$, $P^{ext}_{(:,k)}$
    if $j == 0$ then                                ⇝ first inner iteration in outer iteration
      Update COP: $9.97 - 0.2 \cdot \left(T^{ext}_{(:,k)} - T^{\ell dc}_{(:,k)}\right) + 0.0012 \cdot \left(T^{ext}_{(:,k)} - T^{\ell dc}_{(:,k)}\right)^2$
      Run LP Optimisation Solver
      Calculate mass flux: $\dot{m}^{inj}_{(:,k)} = \frac{\dot{Q}_{(:,k)}}{c_p \Delta T}$                       ⇝ $\Delta T = 5$
      gwhp = GWHP in Hypothetical list with largest $|\dot{Q}_{(:,k)}|$
      Selected.append(gwhp)
      Hypothetical.remove(gwhp)
    for k in All GWHPs do
      if k in Existing-original then
        Update $T^{inj}_{(:,k)} = T^{ext}_{(:,k)} + T^{diff}_{(:,k)}$
      if k in Existing or Selected then
        if $\dot{Q}_{(:,k)} > 0$ then
          Update $T^{inj}_{(:,k)} = T^{ext}_{(:,k)} + 5$
        else
          Update $T^{inj}_{(:,k)} = T^{ext}_{(:,k)} - 5$
    Write Coupling Data to preCICE: $T^{inj}_{(:,k)}$, $\dot{m}^{inj}_{(:,k)}$
    Call preCICE Advance                          ⇝ preCICE controls information flow from here
    if inner iteration converged then
      break

Continue in Algorithm 6.7 to complete the outer iteration step.
```

ALGORITHM 6.6 Pseudo-code of the inner iterations for the urbs adapter solver coupled to preCICE.

inner iterations have converged, the constraint checking is performed as shown in Algorithm 6.7.

The constraint checking procedure requires that each rank communicates globally whether the constraint checking has passed or failed in every region on its own rank. Each rank knows the globally unique region number for every urbs region via the `globalRegionNumber` (first column in Table 6.2). The region number of the downstream urbs region can be obtained from `globalDownStreamRegionNumber` (last column in Table 6.2). Therefore, a list `csv_all` is created of length $N^{urbsG\ell obal}$, which is the total number of urbs regions across all ranks. If the constraints are not satisfied in a region i with the global region number R_i, the `ConstraintPassed` is set to 0 (`False`) in `csv_all` at location `csv_all[`R_i`]`. Initially, `ConstraintPassed` is set to 1 (`True`). An `MPI.AllReduce` step, summing up all values across all ranks, gathers and distributes the sum to all ranks, making a copy of the global constraint checking procedure available on each rank. Each region can inspect if its own downstream region's constraint was satisfied, and if true, the

```
globalRegionNumber[$j$]:  global region number for local urbs region $j$.
globalDownStreamRegionNumber[$j$]:  provides global region number of the
downstream region, for local region $j$
$N^{urbsLocal}$:  Total number of urbs regions on local rank.
$N^{urbsGlobal}$:  Total number of urbs regions across all ranks.
Continue from Algorithm 6.6
if $i == 0$ then                                          ⇝ If end of first outer iteration
  Save natural groundwater temperature state:  $T^{nat}_{(:,k)} = T^{ext}_{(:,k)}$
else
  csv_all = np.zeros($N^{urbsGlobal}$)                ⇝ Create array of length $N^{urbsGlobal}$ on each rank
  for $j = 0, \dots, N^{urbsLocal}$ do                ⇝ Loop through each urbs regions on current rank
    ConstraintPassed = 1 (True)         ⇝ Indicates if constraints are satisfied for urbs region
    if $|T^{nat}_{(:,k)} - T^{ext}_{(:,k)}| > 1$ then                 ⇝ Natural Temperature Constraint
      ConstraintPassed == 0 (False)
    if $P^{inj}_{(:,k)} > P^{inj,nat}_{(:,k)}$ then                          ⇝ Max Pressure Constraint
      ConstraintPassed == 0 (False)
    if $P^{ext}_{(:,k)} < P^{drawdown}_{(:,k)}$ then              ⇝ Minimum Pressure Drawdown Constraint
      ConstraintPassed == 0 (False)
    csv_all[globalRegionNumber[j]] = ConstraintPassed
  csv_all_recv = MPI.AllReduce(csv_all)
  for $j = 0, \dots, N^{urbsLocal}$ do
    ConstraintPassed = csv_all_recv[globalRegionNumber[j]]
    ConstraintPassedDownstream = csv_all_recv[globalDownStreamRegionNumber[j]]
⇝ Check downstream constraint
    if ConstraintPassed == 1 and ConstraintPassedDownstream == 1 then
      Existing.append(Selected_heat_pump)
      Save $\kappa_{min} = \kappa_{max} = \kappa_{gwhp}$         ⇝ Set min and max capacity to the solution of new GWHP
    else
      Removed.append(Selected_heat_pump)
      Save $\kappa_{min} = \kappa_{max}$                  ⇝ Capacity is zero for the GWHP, cannot be selected again
```

ALGORITHM 6.7 Pseudo-code of the outer iterations for the urbs adapter solver coupled to preCICE.

newly added GWHP is moved to the existing list. This procedure is repeated for each region on each rank. Only a single `MPI.AllReduce` command is called once per outer iteration, making this a cheap global communication step.

The implementation of the coupling features for the adapters was described in the algorithms above. However, certain information required for the adapters is not available in the solvers themselves and is read in as additional input files. This is important for two critical parts of the coupling process. First, as the vertex information required to build the coupling surface is not available in the solvers themselves, the mesh must be computed beforehand and read in as an input using the database information. Secondly, the natural groundwater conditions must be stored for each GWHP to ensure that the constraint checking procedure is accurate. Therefore, this information needs to be obtained and formatted into an easily readable format to function with the adapters.

6.2.4 Partitioned Coupling Software Configuration

To perform the simulation coupling, PFLOTRAN and urbs were coupled together via preCICE. Features necessary for partitioned coupling are already available in preCICE, specifically parallel communication, data mapping, implicit coupling schemes and equation coupling acceleration. First, parallel data communication is already implemented. PFLOTRAN and urbs can be decomposed to run in parallel, and preCICE can automatically handle the parallel communication provided that a suitable data mapping scheme is selected. Second, information at each GWHP is exchanged between the solvers. This is implemented as a matching surface interface between PFLOTRAN and urbs, where each GWHP is a vertex on the coupling interface. Third, the implicit coupling scheme defines the inner and outer iteration procedure. Multiple inner iterations are performed within an outer iteration and preCICE checks to see if convergence of the exchanged variables has been reached in each inner iteration.

The urbs adapter can check if the outer iterations (time step in preCICE notation) have converged and, if so, urbs can check the external constraints before moving on to the next outer iteration (coupling time step).

To completely show the required settings, the preCICE configuration file is visualised in Figure 6.9, using the preCICE visualiser tool[8]. In both urbs and PFLOTRAN, a coupling interface is created, called `pflotran_mesh_inj` and `pflotran_mesh_all` in PFLOTRAN, and `urbs_Mesh_inj` and `urbs_Mesh_all` in urbs. The coupling variables pressure and temperature are communicated across these meshes. Both serial-implicit and parallel-implicit schemes are suitable, as the convergence of the coupling variables is required in both cases before the external constraint checking is performed.

6.3 Testing of the Coupling Procedure

The newly developed staggered simulation-optimisation coupling concept has been described in detail. Combined with the novel method for creating the interface coupling mesh to exchange information between the 3D numerical simulation mesh and the optimisation solver graph, testing is required to validate whether:

- the staggered coupling runs sequentially through the regions on each rank in the order specified in Table 6.2,
- the nearest neighbor mapping results in the correct exchange of data between the matching meshes,
- the parallel external constraint checking functions correctly.

For testing, the theoretical example in Section 6.1.3 was recreated in PFLOTRAN and urbs. The simple example has eight hypothetical GWHPs across four urbs regions and a single PFLOTRAN

[8]`https://precice.org/tooling-config-visualization.html`

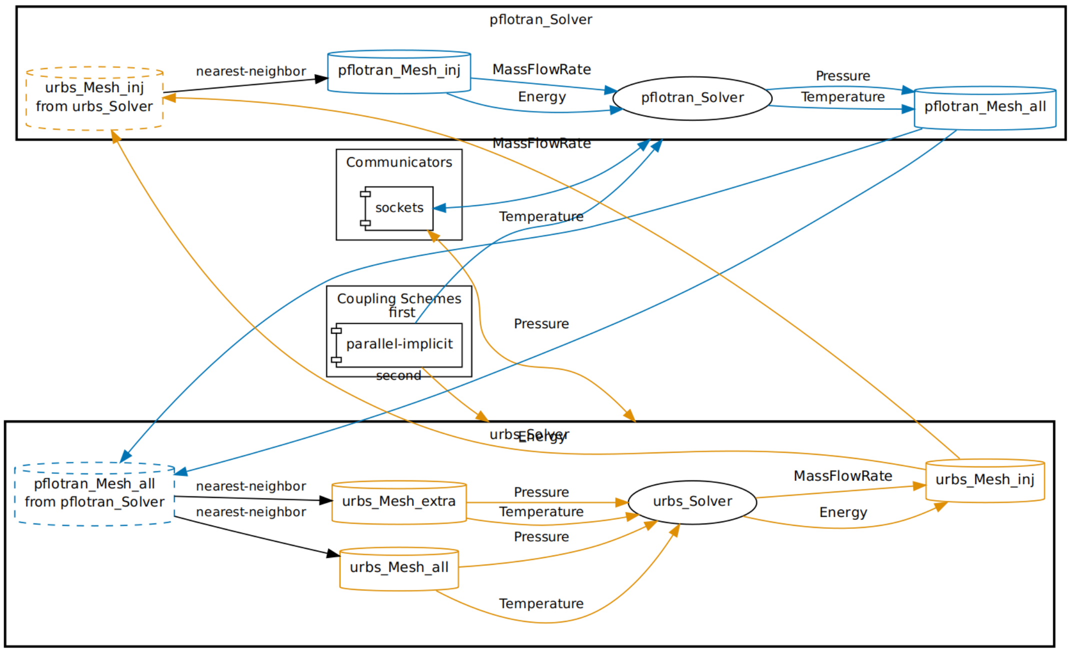


FIGURE 6.9 Visualisation of the preCICE XML configuration for the simulation-optimisation coupling scheme.

region. The PFLOTRAN simulation was run on a single computing rank and the urbs models were run in parallel on two computing ranks. A pressure gradient was applied on the left boundary to induce a fluid flow from left to right in a homogeneous permeability field. A no-flux boundary condition was applied on the top and bottom boundaries to ensure a uni-directional subsurface fluid flow. The simulation was run for 360 days. The results were output at the final time step only, ensuring that the thermal plumes are at the maximum length when constraint checking is performed. The GWHP injection temperatures were set to $T^{inj}_{(:,k)} = T^{ext}_{(:,k)} - 5$, i.e., the thermal plume is at a lower temperature than the background temperature. The modified COP function was set in urbs, being equal to the extraction well temperature for each GWHP, as the lower extraction well temperature reduces the COP and makes the GWHP less efficient. Therefore, the cost-optimal urbs solver would choose the GWHP with the larger extraction well temperature and would not select a GWHP that is already influenced by an upstream GWHP. This test case has two known optimal solutions: *(i)* both GWHPs in region 1, one in region 3 and 1 in region 4 (solution from Section 6.1.3), or *(ii)* one GWHP in each region.

The test case solution is shown in Figure 6.10. The background temperature was set to 10°C with an injection temperature of 5°C. The thermal plume from each injection well (marked with a plus symbol) extends downstream to only one downstream region but interferes with the downstream extraction well (marked with a circle). The solution of the coupled simulation-optimisation test matches the result from Section 6.1.3. The only external constraint check was whether the selected or existing GWHP extraction well had dropped below 9°C, reproducing the

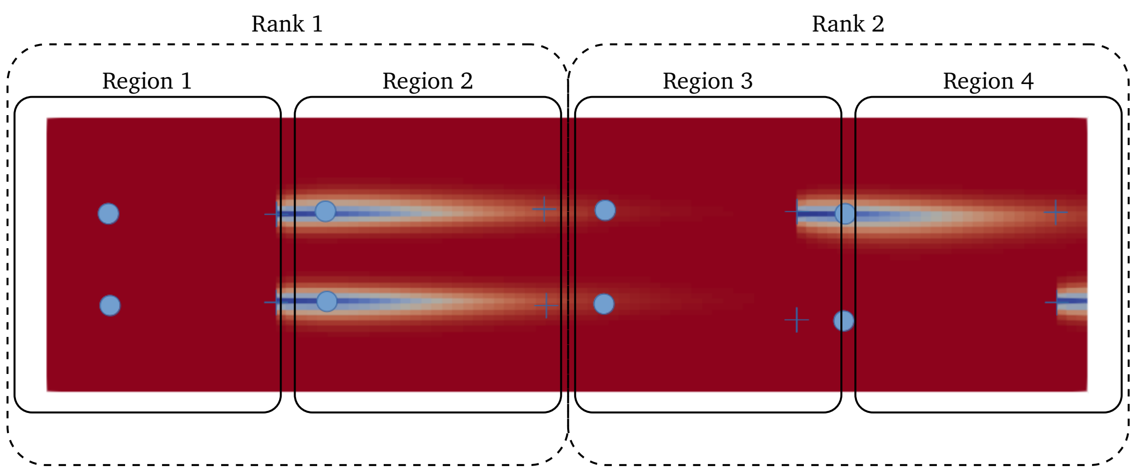


FIGURE 6.10 PFLOTRAN-urbs simulation-optimisation coupling for the four-region model. The known solution is verified with the coupled simulation. The simulation is run for 360 days, and the constraints are checked on the final day only. Each thermal plume extends into one downstream region only, and therefore only influences one downstream GWHP. The computing ranks indicate the urbs region decomposition. The PFLOTRAN simulation was performed on a single computing rank.

constraint that the existing extraction well temperatures cannot change by more than 1°C. This test case proves that the implementation of the nearest neighbor data mapping, external constraint checking and staggered coupling schemes works as expected.

6.4 Summary of Chapter 6

Chapter 6 described the novel coupling procedure that was developed to perform the simulation-optimisation coupling for subsurface geothermal resource optimisation. The requirements for simulation coupling were defined, and the coupling information that each solver requires was described. The parallel-implicit coupling scheme was modified to accommodate the additional external constraint checking, and a staggered coupling scheme was developed to allow for the smaller urbs domains to be run in parallel on distributed systems.

A detailed explanation of the implemented preCICE coupling adapters was provided. The central database providing GWHP information was briefly introduced, explaining how the information required for model generation is accessed. A novel data mapping method was developed using the nearest-neighbor mapping in preCICE with a spatial-temporal mesh, that can map data in both space and time.

Finally, a test case using PFLOTRAN and urbs was generated and tested to validate the new staggered coupling scheme and data mapping method. The theoretically optimal result was achieved, leading the way to performing more complex coupled simulation-optimisation cases in Chapter 7.

7 Analysis of Coupling Schemes for Geothermal Energy Optimisation

The **simulation-optimisation coupling scheme** for geothermal resource optimisation, developed in this thesis, was described in Chapter 6. A staggered coupling approach was developed that allows for multiple urbs models to be run in an alternating fashion and in parallel. To ensure that this approach works as desired, a simple four-region test model was developed and found to replicate the known theoretical solution. However, the purpose of the simulation-optimisation coupling is to run larger PFLOTRAN and urbs models within the city of Munich to provide the cost-optimal usage of GWHPs throughout the city. Therefore, we increase the size of the models to include more complex features in the subsurface domain and a non-structured layout of the urbs regions.

In this chapter, we introduce two new coupled problems: the COM-4 model and the REG-30 model. The COM-4 model is another four-region test case but with more interacting GWHPs than the previous test case and a different urbs region layout. The COM-4 model preparation and the simulation-optimisation results are discussed in Section 7.1, focusing on the correctness of the data mapping procedure. The REG-30 model is a much larger model from the city of Munich that accounts for all the complexities that the developed coupling approach can handle. The REG-30 model preparation and the simulation-optimisation results are discussed in Section 7.2.

7.1 COM-4 Model

The COM-4 model is a slightly more complex example compared to the one presented in Section 6.3. The model describes a simple rectangular domain of a small city area with 14 potential GWHPs, 8 existing-original GWHPs, and 5 subsurface culverts. This includes more complex interactions between existing-original and hypothetical GWHPs, yet is small enough to verify

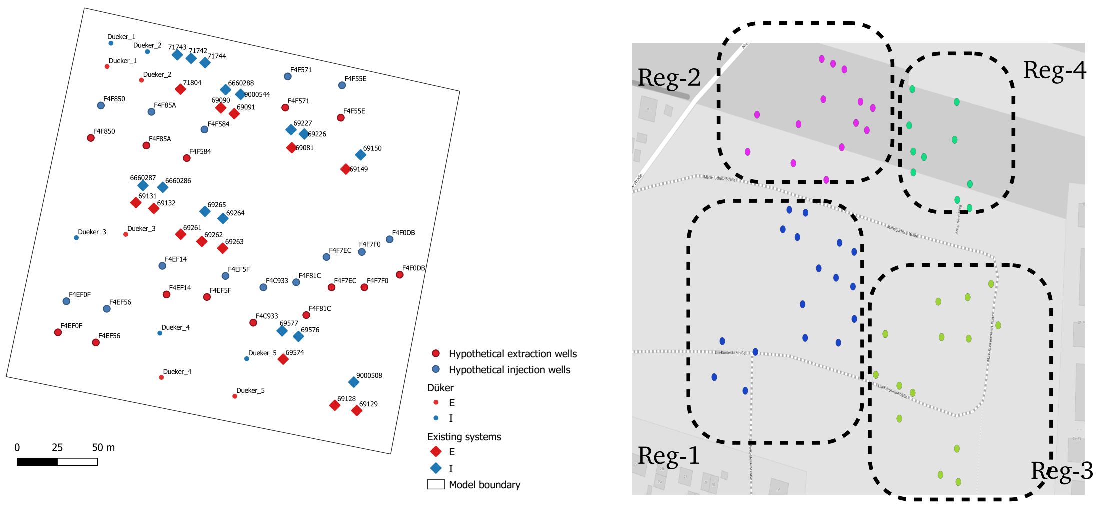


FIGURE 7.1 COM-4 Model: various subsurface infrastructure components (left) and urbs region clustering (right). The subsurface extraction and injection wells are indicated by the red and blue colours, respectively (left). The four urbs regions are clustered into a square layout and coupled to a single PFLOTRAN domain. The groundwater flows from bottom to top, therefore Reg-1 affects Reg-2 and Reg-3 affects Reg-4.

that the PFLOTRAN and urbs adapters function correctly by examining the results of each GWHP individually. The urbs software code and urbs models for each region, software build environment, the PFLOTRAN model and PFLOTRAN software with preCICE adapter are all provided in the DaRUS repository "COM-4 model to replicate simulation results for the GEO.KW project" [Dav22a].

7.1.1 Model Preparation

The layout of the GWHPs through the COM-4 model with the injection and extraction wells of the existing-original GWHPs, the hypothetical GWHPs and the subsurface culverts (referred to as "Dueker"), are shown in Figure 7.1 (left). The domain is divided into 4 separate urbs regions, which are shown in Figure 7.1 (right). Each colour represents the subsurface components within one of the urbs regions Reg-1, Reg-2, Reg-3 and Reg-4. Reg-1 and Reg-2 are run sequentially on the first computing rank Rank$_1$, while Reg-3 and Re-4 are run sequentially on the second computing rank Rank$_2$. Reg-1 and Reg-3 are both run during the same outer iteration, as well as Reg-2 and Reg-4 similar to the example in Section 6.1.3.

The subsurface domain consists of a quaternary and a tertiary layer shown as the top and bottom layer, respectively, in Figure 7.2. The domain has a total of 49 638 polyhedral shaped finite volumes with additional mesh refinement around each GWHP injection and extraction well as well as at each subsurface culvert inlet and outlet. A pressure gradient boundary condition was applied such that the groundwater flows from south to north (bottom to top). The mass flow rate at the existing-original GWHPs was specified in the PFLOTRAN input file.

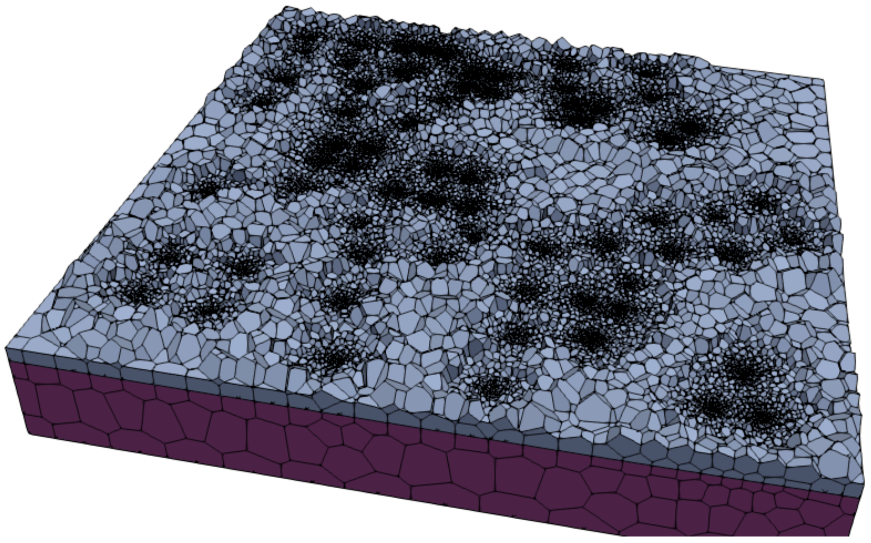


FIGURE 7.2 COM-4 Model: domain mesh of quaternary and tertiary layers, with refinement around the subsurface wells.

7.1.2 Evaluation of the COM-4 Model

There are three key ingredients that need to be validated before larger models can be run: *(i)* data mapping using nearest-neighbor, *(ii)* equation coupling and *(iii)* external constraint checking. The COM-4 model is a relatively simple coupled model similar to that in Section 6.3 but includes all complexities of the real-world model while remaining small enough to manually verify the coupling results. In the following section, we evaluate all three major components of the staggered coupling procedure to ensure that each component has been implemented correctly in the PFLOTRAN and the urbs software adapters (Section 6.2.2 and Section 6.2.3).

To have control over the optimisation input, output and boundary condition values, the optimiser was deactivated for this case. The GWHP selection process was performed by arbitrarily selecting a GWHP from the list of hypothetical GWHPs in each outer iteration. This was done to ensure that a new GWHP was selected in each outer iteration, and that a constant boundary condition with a mass flow rate of $0.6\ell/s$ was specified for each selected hypothetical GWHP to create large thermal plumes and ensure significant interaction between GWHPs, which was used to validate the data mapping step. The injection temperature $T^{inj} = T^{ext} - 5$ was used for all hypothetical GWHPs.

Data Mapping

Purpose: The first test is to verify that values are exchanged correctly between PFLOTRAN and urbs. This is manually done be examining the output values from one solver (e.g., urbs) and verifying that the same input values are set in the other solver (e.g., PFLOTRAN). This is only

```
TemperatureOutput – Rank 0 – Region 1: [0.0, 4.998332372097373, 4.996687224546061,
4.99624852117293, 4.996218035224624, 4.996326799918908, 4.996453386414483,
4.996556489474653, 4.996629367549868, 4.9966772834607625, 4.996707566737152,
4.996726298243745, 4.9967378427834195, ..., 4.996767172722082, 4.996767523668922,
4.996767867245692, 4.996768198293884, 4.996768524330438, 4.996768832515073
```

FIGURE 7.3 COM-4: temperature output from urbs in Region 1 on rank 0. Each value corresponds to the injection temperature at GWHP `F4EF0F` with Cell_{ID} = 49279 at day 0, day 10, day 20 etc.

```
Boundary   Condition   Temperature:     0.0000000000000000   4.9983323720973729
4.9966872245460614   4.9962485211729302   4.9962180352246239   4.9963267999189078
4.9964533864144833   4.9965564894746528   4.9966293675498683   4.9966772834607625
4.9967075667371521 4.9967262982437450 4.9967378427834195 ... 4.9967671727220822
4.9967675236689217   4.9967678672456923   4.9967681982938839   4.9967685243304381
4.9967688325150732
```

FIGURE 7.4 COM-4: temperature input in PFLOTRAN from urbs for the 6th GWHP injection well. Each value corresponds to the injection temperature at day 0, day 10, day 20 etc.

possible due to the relatively small size of the COM-4 model. In the first urbs region, Reg-1, the GWHP $\texttt{Plot}_{ID}$ are listed in the order of GWHPs specified in the urbs input file: `[F4EF0F, F4EF14, F4EF56, F4EF5F, ...]` . As the $\texttt{Cell}_{ID}$ values are required to generate the coupling interface, a database similar to Table 6.3 can be used to find the injection and extraction well $\texttt{Cell}_{ID}$ value for each $\texttt{Plot}_{ID}$. For this specific model, the values of the injection well $\texttt{Cell}_{ID}$ are: `[49279, 36795, 46985, ...]` . Examining the PFLOTRAN input file, $\texttt{Cell}_{ID} = 49279$ is associated with the injection well of the 6th GWHP listed in the PFLOTRAN input file `pflotran.in` . Therefore, the output values from the first GWHP in the urbs model in Reg-1 must be mapped to the 6th GWHP in PFLOTRAN.

Results: The temperature output for the GWHP `F4EF0F` at Cell_{ID} 49279 in the urbs adapter is shown in Figure 7.3, followed by the input boundary condition temperature for the 6th GWHP in the PFLOTRAN boundary condition list in Figure 7.4. The values in both figures are specified for $\text{Cell}_{ID} = 49279$ in 10 day intervals.

Discussion: Ensuring that the correct temperature, pressure and mass flow rate at the correct GWHP, and at the correct point in time are exchanged between PFLOTRAN and urbs is critical for the partitioned coupling scheme. The developed method of utilising the $\texttt{Cell}_{ID}$ value with a time stamp allows for the correct exchange of data when combined with the consistent nearest neighbor mapping (Section 2.3.1), as shown in Figure 7.3 and Figure 7.4. The values displayed in each figure are obtained from the output log files from urbs and PFLOTRAN. In Figure 7.3, the order of the temperature values for GWHP `F4EF0F` with $\texttt{Cell}_{ID} = 49279$ is sequential in time (day 0, day 10, day 20, etc.). When applying the boundary condition in PFLOTRAN, the values are also read in sequentially (day 0, day 10, day 20, etc.). Therefore, it is possible to manually check the boundary condition values for each GWHP. The manual verification of the mapping values was performed for all coupling data for each GWHP in both directions (urbs to PFLOTRAN and PFLOTRAN to urbs). For brevity, we limit the verification results to a single example shown above. The data mapping method works as expected, even when the urbs regions and PFLOTRAN domains are decomposed across multiple ranks, due to the parallelised nearest-neighbor mapping functionality in preCICE.

Equation Coupling

Purpose: Equation coupling acceleration using quasi-Newton methods (see Section 2.2) aims to increase the convergence rate for partitioned coupled problems. The purpose of this test is to observe whether quasi-Newton methods reduce the number of inner iterations when compared to exchanging the values directly between PFLOTRAN and urbs using a constant under-relaxation with the under-relaxation factor equal to one (Equation 2.7 with $\omega = 1$). The IQN-ILS method was used as described in Section 2.2.2, with 10 outer iterations (time steps in preCICE notation) reused and no filtering or pre-scaling. Pre-scaling was not required as only the urbs and PFLOTRAN temperature values were used for the quasi-Newton vector and are therefore in the same order of magnitude range. The convergence threshold was varied for both direct exchange and quasi-Newton methods, as a lower threshold makes it harder for either method to converge.

Results: The total number of coupling iterations for the COM-4 model over the first 10 outer iterations for a convergence threshold of $\epsilon_{conv} = 10^{-4}$ and $\epsilon_{conv} = 10^{-5}$ is shown in Table 7.1.

Discussion: The quasi-Newton coupling method was able to reduce the number of inner iterations by 3 and 2 iterations for the convergence thresholds of $\epsilon_{conv} = 10^{-4}$ and $\epsilon_{conv} = 10^{-5}$, respectively. This is only a minor improvement with the additional quasi-Newton expense. In the COM-4 test case, there is a significant amount of interaction between the GWHPs due to the high mass flow rate of the hypothetical GWHPs. This tests the limits of the equation coupling and constraint checking procedure. The impact that a thermal plume has on a downstream GWHP is only obtained after an inner iteration is complete. In each inner iteration, PFLOTRAN receives updated injection well temperatures from urbs. In an example case, a previously inactive GWHP is activated and a mass flow rate is assigned to it and sent to PFLOTRAN. After completing a full PFLOTRAN simulation, the updated extraction well temperatures are sent to urbs. If a downstream GWHP extraction well temperature is influenced by the newly activated upstream GWHP, the injection well temperature of the downstream GWHP is only updated in the following inner iteration. This cycle may repeat several times if there are many interacting thermal plumes.

TABLE 7.1 COM-4: Total number of inner coupling iterations comparing the direct information exchange of coupling variables ("Constant") versus the quasi-Newton acceleration ("QN") over 10 outer iterations. The test was repeated for convergence thresholds of $\epsilon_{conv} = 10^{-4}$ and $\epsilon_{conv} = 10^{-5}$.

$\epsilon_{conv} = 10^{-4}$		$\epsilon_{conv} = 10^{-5}$	
Constant	QN	Constant	QN
49	46	52	50

Therefore, without this "chain of plumes" appearing after the first inner iteration, the quasi-Newton method will not be able to model or account for this behaviour as it did not exist before.

In the opposite case where there is limited interference between GWHPs, changing the mass flow rate or temperature at certain injection wells might not change the temperature at any downstream extraction wells at all. This will always result in fast convergence of the inner iterations, as any injection well temperature input in PFLOTRAN will have little to no impact on the extraction well temperatures sent to urbs and, subsequently, the input coupling data in PFLOTRAN will remain unchanged. In the limited interaction scenario, using constant under-relaxation could have equal performance to the quasi-Newton method then. Therefore, the quasi-Newton method may only have a limited advantage for a small number GWHP with a moderate amount of interference, but without forming a long chain of interacting plumes that require numerous inner iterations before the complete interaction of all GWHPs is apparent.

Constraint Checking

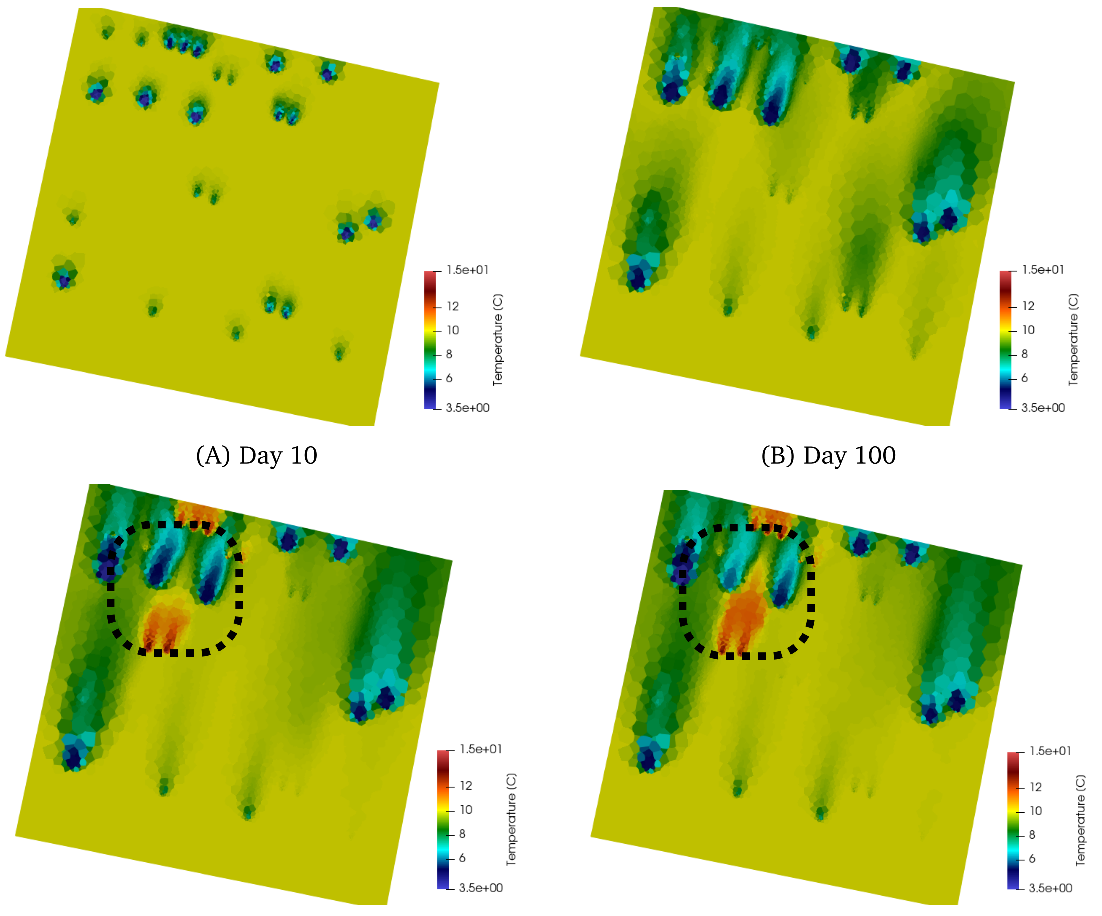
Purpose: At the end of each outer iteration, the constraints are checked to ensure that all GWHPs operate within a specified operating range. The final test ensures that no hypothetical GWHP is moved to existing list if the constraints are not met. The specific set of constraints were set to:

- Existing-original: the extraction well temperature must not change by more than 1°C from the natural temperature. The natural temperature at each extraction well was set to the extraction well temperature at the end of the first outer iteration, when no hypothetical GWHP was selected yet. This provides the baseline thermal field.
- Hypothetical: the temperature at the injection well[1] must not be less than 4°C or more than 20°C, otherwise the GWHP is removed.

Results: The subsurface temperatures for the COM-4 model of the optimised solution after 10 outer iterations are shown at day 10, 100, 200 and 250 in Figure 7.5. The temperatures are shown at a slice 482 meters above sea level, where the thermal plumes of all wells can be observed. The simulation begins at the start of January at day 0 during winter, proceeds through

[1]The injected water mixes with the surrounding groundwater, therefore the groundwater itself does not necessarily reach 4°C.

the summer and continues to day 360 in December. The groundwater temperature colour scaling is plotted in the colour bar for each sub-figure. The isovolumes of the thermal plumes between 4.4°C and 8°C at day 100, against a background temperature of 10°C, are shown in Figure 7.6.



(A) Day 10 (B) Day 100

(C) Day 200 (D) Day 250

FIGURE 7.5 COM-4: Subsurface temperatures for the optimised solutions, satisfying all constraints, at day 10, 100, 200 and 250. The image is sliced at 482 meters above sea level. The simulation starts at the start of January on day 0 during the winter months, and runs until day 360 in December

Discussion: The simulation-optimisation coupling of the COM-4 results in only 8 of the 14 hypothetical GWHPs being adding to the final solution. Both constraint violations were the cause of removing GWHPs: changing of the existing-original extraction well temperature and the injection well of hypothetical GWHPs temperature extending outside the allowed range.

The existing-original GWHPs function as either heating or cooling devices depending on the season. At the start of the year at day 10, the existing-original GWHPs inject colder water into the ground as energy is extracted from the groundwater to heat buildings in the winter. Around day 200, the existing-original GWHPs are used to cool buildings instead, which heats up the groundwater. However, the hypothetical GWHPs did not account for a change in season and always used the criteria $T^{inj} = T^{ext} - 5$ as a boundary condition, resulting in a mixed usage of the subsurface aquifer.

The interaction between GWHPs over time can be clearly seen between day 200 and day 250 as marked in the dashed box in Figure 7.5. As the warm (red) thermal plume extends downstream (upwards in the figure) between day 200 to day 250, the plumes reach the extraction wells of the downstream heat pumps. The warmer extraction well temperatures result in warmer injection well temperatures (lighter blue), as noted at day 250. Additionally, the warmer subsurface water mixes with the downstream injection well fluid in the subsurface.

The isovolume surfaces in Figure 7.6 provide a 3D perspective of the thermal plumes and show how the plume extends not only downstream in plane, but vertically up and down. The thermal plumes of the hypothetical GWHPs are much larger than those of the existing-original GWHPs. This is due to the larger mass flow rate of the hypothetical GWHPs. The existing-original GWHPs still achieve a low temperature at the injection well itself, however it quickly mixes with the surrounding groundwater and, therefore, the downstream temperature is not affected as much as by the new GWHP.

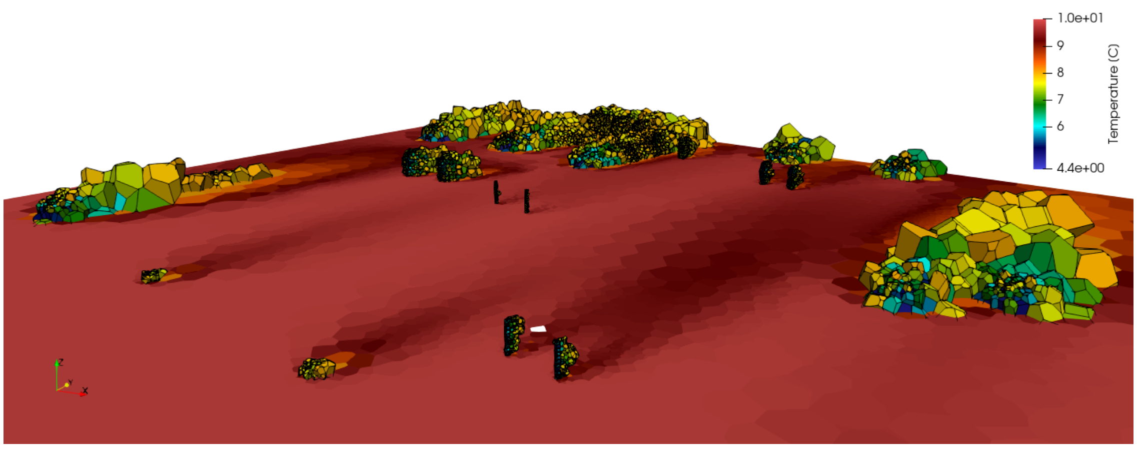


FIGURE 7.6 COM-4: isovolume surfaces of thermal plumes that are between 4.4°C and 8°C for the optimised solution at day 100. The isovolume surfaces provide a 3D representation of the thermal plume extending in all dimensions.

7.2 REG-30 Model

The COM-4 model was ideally suited to test the functioning of the developed partitioned coupling procedure. The staggered coupling scheme, data mapping and constraint checking procedure could be manually validated with minimal effort, while considering all complexities of a larger model. The REG-30 model is much larger than the COM-4 model and covers a large area in the south-east part of Munich as shown previously as number 30 in Figure 6.3 (left). It contains 800 hypothetical and 31 existing-original GWHPs, with 37 and 43 injection and extraction wells for the existing-original GWHPs. Additionally, the PFLOTRAN domain contains above surface water bodies, subsurface culverts and tunnels. The existing-original systems consist of only heating, only cooling, and combined heating and cooling systems, as shown in Figure 7.7 (left). In this section, the model generation process is described, followed by the simulation-optimisation results. The urbs software code and urbs models for each region, software build environment, the PFLOTRAN model and PFLOTRAN software with preCICE adapter are all provided in the DaRUS repository "REG-30 model to replicate simulation results for the GEO.KW project" [Dav22b].

7.2.1 Model Preparation

The model construction and setup require careful attention and was prepared by experts in their respective fields. The PFLOTRAN model was generated by the Chair of Hydrogeology at the Technical University of Munich. The urbs models and urbs region clustering was performed by the Chair of Renewable Energy Systems at the Technical University of Munich.

PFLOTRAN

The first step in generating the PFLOTRAN model of a single region is to create a domain with the necessary features such as GWHP wells, surface water bodies and shallow subsurface structures such as culverts or tunnels. Each hypothetical GWHP contains only one injection and one extraction well, where the locations are shown in Figure 6.3 (right). Once the domain geometry is generated, a tetrahedral mesh is created with additional mesh refinement around the GWHPs to improve the local numerical solution around each GWHP.

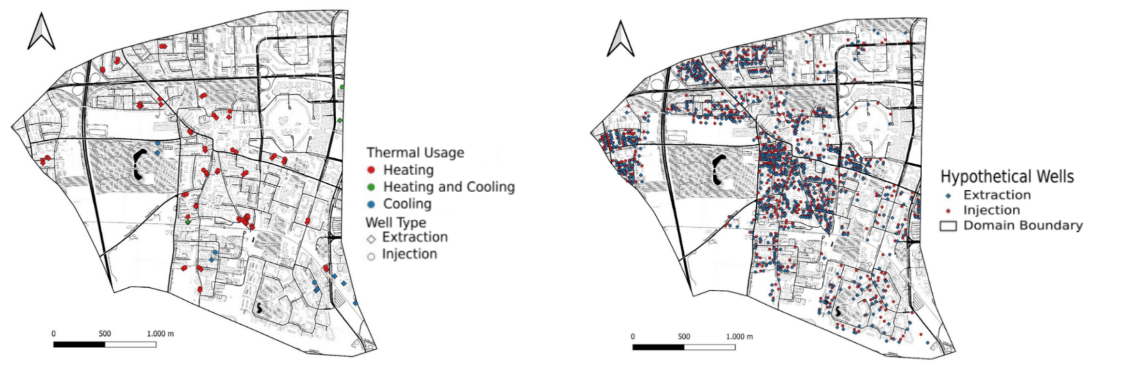


FIGURE 7.7 REG-30 domain with existing heating and cooling systems (left) and hypothetical heat pump locations (right).

The tetrahedral mesh is converted into a polyhedral mesh by fusing adjacent tetrahedra, improving the mesh quality and reducing the mesh size to 4 034 584 finite volume elements

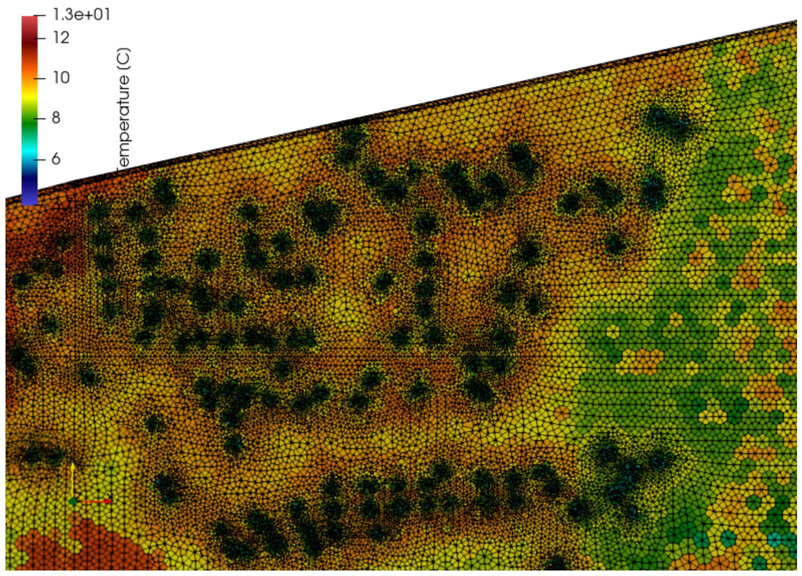


FIGURE 7.8 REG-30: polyhedral mesh of the north-west area of the PFLOTRAN domain with a high density of GWHPs. The refinement of the mesh around each GWHP accounts for the large temperature and pressure gradient around each GWHP, but also increases the number of elements in the mesh.

for the REG-30 model. The number of finite volume elements must remain low as each inner iteration requires the full solution of the PFLOTRAN model. The polyhedral mesh in the north-west region of the domain is shown in Figure 7.8, highlighting the refinement around each GWHP well location. The simulation runtime was set from day $t_{start} = 0$ until day $t_{end} = 360$ with a simulation time step $\Delta t^{sim} = 5$ days, where the values on the coupling interface are output every 10 days and a VTK output file for visualisation purposes[2] was written every 60 days. A pressure gradient boundary condition is applied at the southern and northern boundaries to induce the flow of groundwater through the domain. A zero flux condition was applied on the east and west boundaries. The known mass flow rate was specified for each existing-original GWHP, a zero mass flow rate for each hypothetical GWHP. No temperature boundary condition was applied at the extraction wells and a temperature of $T^{inj} = 10$ for all hypothetical GWHP injection wells.

urbs

There are two steps required to generate the urbs input files: *(i)* the region clustering to decompose the region into multiple urbs sub-domains and *(ii)* well-summary pre-processing to assign the $\texttt{Cell}_{ID}$ (obtained during the PFLOTRAN region meshing) to each injection and extraction well. The locations of all existing-original and hypothetical GWHPs are supplied by the main database. Combined with the known Darcy velocity streamlines throughout the entire city, multiple urbs regions are generated during the region clustering process, as described in Section 6.1.3.

[2]The VTK output is not required for simulation coupling.

TABLE 7.2 REG-30: conversion process investment ("Inv. Cost"), fixed ("Fix. Cost") and variable ("Var. Cost") costs for the urbs models. The GWHP investment costs vary depending on the location and building type.

Conversion Process	Inv. Cost [€ /MW]	Fix. Cost [€ /MW a]	Var. Cost [€ /MWh]
	Heating Infrastructure		
Oil Heating	639,302.52	23,470.76	4.62
Gas Heating	799,617.65	35,498.44	0.79
GWHP	24,768.00 – 1,266,502.41	41,523.85	52.90
Air Heat Pump	1,500,000.00	5,000.00	–
District Heating	366,044.00	57,791.32	–
Pellet Heating	1,588,963.26	44,822.16	0.70
Solar Thermal	1,000,000.00	–	
	Cooling Infrastructure		
Air Heat Pump	166,095.00	41,524.00	52.9
Air-conditioning	166,095.00	41,524.00	52.9
	Electrical Infrastructure		
Photovoltaic	1,309,000.00	39,270.00	–
Electrical Supply	–	300.00	30.0

After the urbs model generation and clustering was finalised, a pre-processing step was performed to couple the GWHP $\texttt{Plot}_{ID}$ information with the $\texttt{Cell}_{ID}$ information. This pre-processing step was developed during the development of the coupling adapters in this thesis. The $\texttt{Plot}_{ID}$ of each GWHP is checked in the database, which returns the $\texttt{Cell}_{ID}$, for both existing-original or hypothetical GWHPs, and the minimum and maximum groundwater pressure at each well. For any existing-original system, the known temperature difference between the injection and extraction well and the extraction well mass flow rate are returned to the pre-processing adapter. After clustering and pre-processing, the urbs region consists of 26 urbs models that run on 13 computing ranks.

The cost-optimal solution is highly dependent on the cost of each conversion process that can be used to meet the space heating and cooling demands. The specific investment, fixed and variable costs for each conversion process is provided in Table

7.2.2 Calibration

Once the mesh has been created along with the input configuration file for the PFLOTRAN model, including all boundary conditions, the groundwater calibration is performed. The calibration of the groundwater hydraulic permeability field was performed as described in Section 5.3.3. A set of pilot points were distributed throughout the domain in both the quaternary and tertiary layers (large dark green dots) and with additional pilot points placed in the quaternary layer (smaller light green dots) in Figure 7.9. The tertiary layer is less permeable than the quaternary layer,

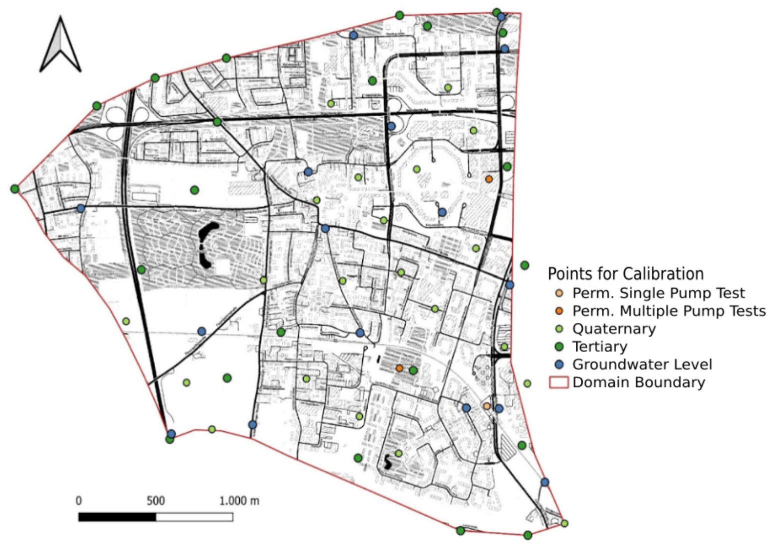


FIGURE 7.9 REG-30: locations of pilot points and real-world measurement positions in both the quaternary and tertiary domain. The permeability value set at each pilot point is interpolated to 4 034 584 finite volume elements. The real-world observation data are the single pump and multi-pump permeability obtained from pumping tests and the groundwater level measurements.

and therefore does not need to be resolved in as much detail as the quaternary layer. The known permeability field values, measured from both single well and multiple well pumping tests (light and dark orange dots), and groundwater level measurements were used to calibrate the model.

During the calibration procedure, PEST++ determines the permeability value at each pilot point, which is then mapped to the finite volume mesh using the radial basis function interpolation method described in Section 5.3.3 using Algorithm 5.1. The permeability value at each pilot point is adjusted until the error between the mapped permeability field and the known permeability value is minimised, along with reducing the error between the measured groundwater level and the simulation groundwater level observations.

The final permeability field in the quaternary and tertiary layer are shown in Figure 7.10. The quaternary layer permeability field values are approximately 4 to 5 orders of magnitude larger than those in the tertiary layer, which allows for a higher groundwater velocity in the quaternary layer. With the models for region 30 of the city of Munich complete, the simulation-optimisation coupling can be performed.

7.2.3 Evaluation of the REG-30 Model

The results from the numerical experiments of the REG-30 model are presented below. During an initial coupled simulation-optimisation run, the data mapping procedure was checked for a handful of GWHPs to ensure that the data mapping procedure worked correctly but was not

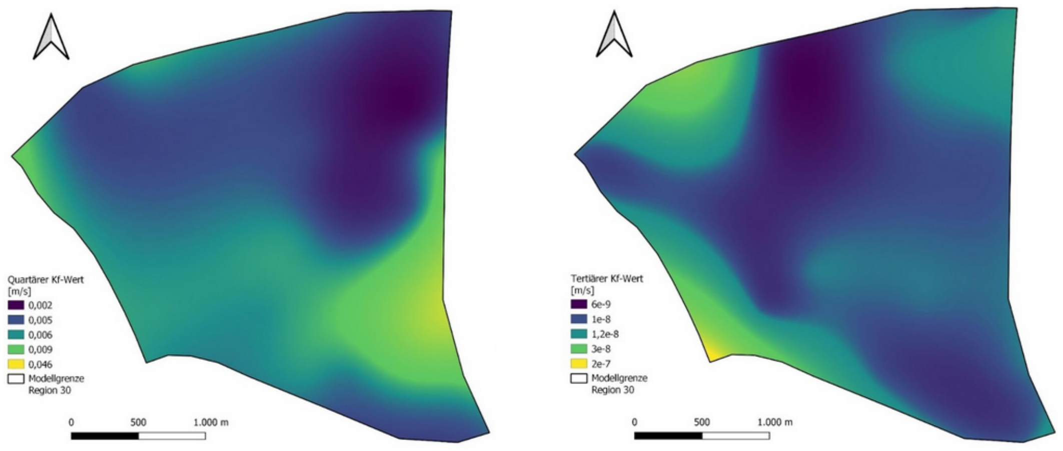


FIGURE 7.10 REG-30: final permeability field in the quaternary (left) and tertiary (right) layers. The permeability values in the quaternary layer are orders of magnitude larger than the tertiary layer.

performed for all 800 hypothetical and 31 existing-original GWHPs. The numerical experiments were divided into the following tests: *(i)* scalability testing of PFLOTRAN and urbs, *(ii)* equation coupling acceleration and *(iii)* coupled simulation-optimisation tests.

All simulations were performed on SuperMUC-NG at the Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities[3]. The tests were performed on the "general" partition nodes, consisting of Intel Xeon ("Skylake") processors with a total of 6336 nodes with 48 cores and 96 GB RAM per node.

Scalability

Purpose: The purpose of the scalability tests is:

- to determine the runtime behaviour of PFLOTRAN when using more computing resources and
- to determine the runtime behaviour of urbs when using more GWHPs per urbs model and with finer optimisation time step sizes.

The PFLOTRAN model for each region can be decomposed over a larger number of MPI ranks using standard internal domain partitioning to reduce the simulation runtime, provided that PFLOTRAN has good scaling behaviour. The urbs models cannot simply be decomposed over a larger number of domains after the domain decomposition and region clustering is complete. Therefore, it is necessary to know how the urbs models scale with increasing/decreasing number of conversion processes (Section 5.4.2) and GWHPs per urbs region. However, the urbs optimisation time step can be increased, i.e., the energy demand and optimisation solution can be specified hourly, daily

[3]`https://doku.lrz.de/display/PUBLIC/High+Performance+Computing`

TABLE 7.3 REG-30: Scalability results for the PFLOTRAN model showing the runtime in seconds for various solver components with varying number of computing ranks. The iterative solver runtime ("Sim Time") was calculated by subtracting the VTK file output ("VTK") from the total solver runtime ("Total Time").

	Rank							
	48	96	192	384	768	1536	3072	6144
Sim Time	2187.36	1101.92	555.89	297.09	154	80.45	51.39	24.77
VTK	0.64	4.68	9.51	12.41	19.5	27.75	38.61	80.73
Total Time	2188	1106.6	565.4	309.5	173.5	108.2	90	105.5

or weekly (refinement in time and not in space), to reduce the LP solver time.

Results: The scaling results for PFLOTRAN are provided in Table 7.3, listing the runtime of the iterative solver in PFLOTRAN ("Sim Time"), the time to write the output to VTK files for visualisation ("VTK"), and the total simulation time ("Total Time"). The "Sim Time" was calculated by subtracting the "VTK" time from the total simulation runtime. The number of computing ranks (cores) was increased from 48 to 6144. Each PFLOTRAN scaling test was performed by running a single inner iteration high-fidelity simulation from day 0 to day 360, with VTK output every 120 days (a total of 4 VTK output files produced at days 0, 120, 240 and 360).

The scaling test for the urbs models are shown in Table 7.4, showing the runtime in seconds for various components of the urbs optimisation solver (Gurobi, Sel. GWHP, Energy Rate and Total) for different urbs regions. In each urbs region (row "Region"), the number of GWHPs within that region is provided (row "GWHPs"). The data set is split into two, one where the optimal solution is solved on a daily resolution (middle row block), and one on a weekly resolution (bottom row block). Each row block contains four optimisation components: *(i)* Gurobi, which is the software used to solve the LP problem, *(ii)* Sel. GWHP, which is the function that determines which GWHP is selected by the LP solution, *(iii)* Energy Rate, which is the function that determines the energy flux/energy rate through each GWHP and is used to determine the mass flow rate in each GWHP and *(iv)* Total, which is the sum of the previous three measurements.

Discussion: Evaluating the PFLOTRAN results in Table 7.3, it is observed that the "Sim Time" scales well with increasing the number of ranks. Even when doubling the number of ranks from 3072 to 6144, the simulation solver time is less than half. This is due to some measurement error for the simulation timing, the total runtime is only reduced from 80.45 to 24.77 when increasing the number of ranks by a factor of 4 (highlighted in the table). However, this scaling result is excellent for the number of ranks and results in a fast simulation time. The VTK file output time increases when increasing the number of computing ranks as this involves a global MPI step to concatenate the data from all ranks onto a single main rank. This ruins the scaling performance of PFLOTRAN, where 76% of the simulation time is dedicated to generating the VTK output files on

TABLE 7.4 REG-30: Runtime in seconds for various components of urbs for daily and weekly optimisation time resolution, with varying number of GWHPs per urbs region. The urbs region number and number of GWHPS per region are provided in the top two rows. The runtime of the LP solver ("Gurobi"), the GWHP selection function ("Select GWHP"), the energy rate and mass flow rate calculation ("Energy Rate") and combined total of all three ("Total") are provided for two different optimisation time resolutions: daily (middle row block) and weekly (bottom row block).

	Region	1	3	4	1	6	7	10	9
	GWHPs	11	16	22	24	27	28	40	44
Daily	Gurobi	51.69	72.45	102.51	113.78	132.21	132.82	189.09	202.08
	Select GWHP	2.29	5.03	2.29	12.25	14.04	16.54	33.0	37.75
	Energy Rate	4.65	10.28	20.35	24.99	31.73	33.67	73.38	87.53
	Total	58.62	87.76	125.15	151.02	177.98	183.03	295.47	327.36
Weekly	Gurobi	15.82	10.46	15.51	7.07	19.32	18.64	29.00	32.25
	Select GWHP	1.38	1.79	1.14	0.28	1.79	1.69	4.18	5.18
	Energy Rate	2.96	3.97	2.39	0.57	3.64	3.73	7.92	9.82
	Total	20.16	16.22	19.04	7.92	24.75	24.06	41.11	47.24

6144 ranks. However, fast simulation coupling is possible as the VTK output is not a requirement to perform the simulation-optimisation coupling.

Evaluating the urbs results in Table 7.4, "Gurobi" is the runtime that the LP solver Gurobi requires to solve the LP problem, which appears to increase linearly with increasing number of GWHPs. However, the "Select GWHP" and "Energy Rate" runtimes have a worse than linear scaling with the number of GWHPs. Both functions loop through all GWHPs in the urbs model to determine whether each GWHP has a non-zero energy usage. They could potentially be combined into a single function to reduce the overhead. This will be implemented in future versions of urbs to reduce the coupling runtime. Due to the almost linear scaling of runtime with the number of GWHPs in each region, each region should have a similar number of GWHPs to improve the load balancing between ranks.

The runtime of all four components reduces significantly when reducing the optimisation time resolution from daily to weekly. This comes at the expense of the accuracy of the optimised solution. However, the groundwater dynamics are slow and the extraction well temperatures do not undergo large changes from day to day, but only vary significantly between seasons. Therefore, using a weekly time resolution still captures the effects of moving between heating and cooling modes of the GWHPs. This results in an adequate solution for the energy usage of GWHPs and realistic groundwater plume sizes, while remaining at a reasonable computational cost.

Equation Coupling Acceleration

Purpose: The numerical tests of the COM-4 model revealed that the quasi-Newton coupling acceleration offered a minimal improvement over using constant under-relaxation (direct ex-

TABLE 7.5 REG-30: Number of inner iterations per outer iteration (Out. Iter.) for the IQN-ILS coupling and direct exchange of data. Due to the model size, the simulations were terminated after 60 minutes.

Out. Iter.	1	2	3	4	5	6	7	8
Direct	7	3	3	3	3	3	3	3
QN	10	7	5	6	—	—	—	—

change). In the scenario where a long chain of interacting plumes might be present, the influence that each plume has on another GWHP would require a complete inner iteration to be performed before the effect is observed in the following inner iteration. On the other hand, if the model does not have any interaction between plumes and GWHPs, the direct exchange (a plain fixed-point iteration) of information might perform just as well as quasi-Newton coupling as the PFLOTRAN output does not change as the usage of GWHPs changes. The quasi-Newton equation coupling and direct exchange of data are compared for the REG-30 model. The IQN-ILS quasi-Newton method was selected, with a convergence threshold of $\epsilon_{conv} = 10^{-3}$, without filtering or pre-scaling, and with 10 outer iterations (`time-steps-reused` in preCICE notation) and 100 `max-iterations-reused`. The key criterion for evaluating the effectiveness of the equation coupling acceleration is to reduce the number of coupling iterations.

Results: The total number of inner iterations per outer iteration ("Out. Iter.") is shown in Table 7.5 for both the direct exchange of data ("Direct") using a constant under-relaxation with ω = 1 (Equation 2.7, also a plain fixed-point operation) and the quasi-Newton method ("QN"). The coupled simulation was terminated after 60 minutes to save computational resources performing multiple tests. No VTK output was performed.

Discussion: The direct exchange of data outperformed the quasi-Newton coupling by a significant margin for the REG-30 model. For the direct coupling, a total of 7 inner iterations were required for the first outer iteration, followed by 3 inner iterations per outer iteration. This indicates that there are few, if any, thermal plume chains forming due to multiple interacting GWHPs. The quasi-Newton coupling suffers from reduced performance, only completing 4 outer iterations in the same time that the direct exchange completes 8. For smaller convergence thresholds of $\epsilon_{conv} = 10^{-1}$, the direct exchange and quasi-Newton method require the same number of inner iterations per outer iteration.

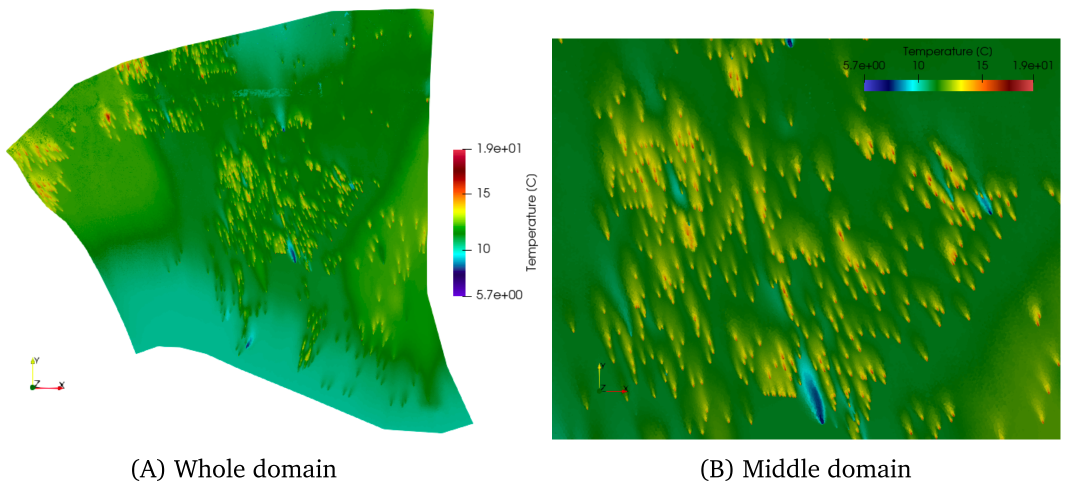
A second disadvantage of the quasi-Newton coupling is that the values at the coupling interface are not exact and are instead determined by the quasi-Newton update step to minimise the interface coupling residual. Unless the convergence threshold is set very low, the temperatures and pressures in one solver might not match the values in the other. Due to the additional external constraint checking required in urbs, the direct exchange of data is a favourable option for the equation coupling due to the fast convergence rate and exact matching of the interface values between the solvers.

Optimised Result

Purpose: The final numerical experiment is to perform the full scale simulation-optimisation coupling for the REG-30 model. The coupled simulation-optimisation technique offers a unique advantage over non-coupled methods, as the thermal impact of the operational GWHPs is included in the optimised solution. In the following test, we evaluate the quality of the coupled simulation-optimisation solution and not the objective function value of the urbs optimisation. To provide context to the staggered simulation-optimisation coupling, various alternative optimisation objectives are compared.

1. Baseline: the simulation-optimisation coupling was performed with an artificial mass flow rate of $\dot{m} = 0.1\ell/s$ for all hypothetical GWHPs in the domain to visualise the extent of the interaction between GWHPs if they are all active. This provides an estimate of the impact of using all hypothetical GWHPs, irrespective of the constraints or cost-effectiveness.
2. Non-Coupled: a non-coupled optimisation is performed that does not account for mutual interaction between PFLOTRAN and urbs. Multiple inner iterations are performed in the first outer iteration until convergence of the existing-original GWHPs is attained. Next, the optimisation is performed in each region using the temperatures at the extraction wells to determine the COP for each GWHP. No limitation is placed on the number of hypothetical GWHPs that can be selected as no constraints are checked. This is equivalent to providing each urbs model with the extraction well temperatures from an initial solution and solving each urbs model independently.
3. Single-Region-Optimisation performs the normal, staggered coupling approach as described in Section 6.1.3, where the LP problem is solved for one region per rank in each outer iteration.
4. Double-Region-Optimisation the LP problem is solved for every region on each rank per outer iteration. This may cause more GWHPs to be removed, as if the constraint for a GWHP is violated, at least two GWHPs are removed from the final optimised solution (either the GWHP added in the current region or in the upstream region may cause the constraint violation, therefore both are removed). This would speed up the optimisation process as fewer outer iterations are required, at the expense of more GWHPs being removed.
5. Zero-Cost: The investment, fixed and variable costs ζ_{inv}, ζ_{fix} and ζ_{var}, respectively, are set to zero for each GWHP. This forces the LP solver to select a GWHP in each outer iteration when using the cost-optimal objective function.

All tests utilise a minimal cost objective function as described in Section 5.4.3. Only the selection of the GWHPs is analysed and not the quality of the objective function. The purpose is to show that the technical simulation coupling, data exchange and constraint checking functions correctly. In each test, the two key features to observe are: *(i)* how many GWHPs are added to the



(A) Whole domain (B) Middle domain

FIGURE 7.11 REG-30: groundwater temperatures for the baseline simulation case with all existing-original and hypothetical GWHPs active. The temperature on day 120 is displayed for the whole domain (left) and zoomed into the middle of the domain (right) with significant GWHP interaction. The temperatures are displayed on a slice at 525m above sea level.

existing set and *(ii)* how many GWHPs are removed for each region. If a region has only a small number of GWHPs added to the existing set, but also has a small number of removed GWHPs, this means that no GWHPs are being selected in the optimisation, and that the GWHPs are not a feasible means of meeting the heating and cooling demands.

Results: The result for the baseline test, where all hypothetical GWHPs are active, is shown in Figure 7.11. The temperatures are displayed at day 120 and at a slice generated at 525m above sea level. The temperatures for the whole domain are shown on the left, where an enlarged image of the centre of the domain is displayed on the right. Each hypothetical GWHP operates with $\dot{m} = 0.1\ell/s$.

The groundwater temperatures for the Single-Region-Optimisation scenario are displayed in Figure 7.12 at day 0, 60, 120, 180, 240 and 300. The Single-Region-Optimisation utilised the staggered coupling approach developed in Section 6.1.3. The groundwater temperatures at day 0 begins with a semi steady-state solution by running the simulation with the existing-original GWHPs over a period of 2 years. The temperature range is displayed individually to the right of each sub-figure.

The number of GWHPs that are selected and removed during the simulation-optimisation coupling process is shown in Table 7.6. The region number is displayed in the left hand column. The total number of existing-original GWHPs in each urbs region is shown at the start of the coupling simulation-optimisation procedure under Start – "Exis.", with the total number of hypothetical GWHPs to select from under "Hyp.". The total number of existing and removed GWHPs in each urbs region at the end of each coupling scenario is provided under "Exis." and "Rem.".

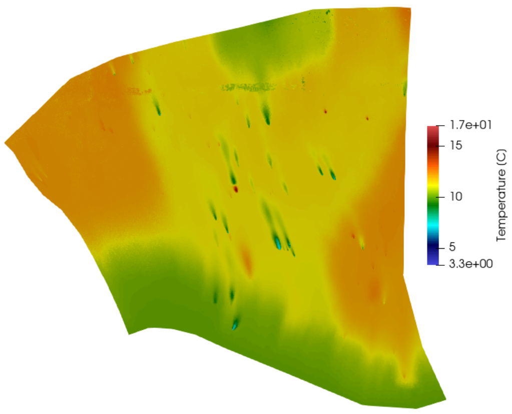
A comparison of the temperature at the centre of the REG-30 model is shown in Figure 7.13 for the test case where each GWHP has zero installation, fixed and variable cost (left) and with regular cost (right).

Discussion: Examining Figure 7.11, the impact of using all hypothetical GWHPs within the domain is shown at day 120. Even with the significant interaction of the thermal plumes, the minimum temperature constraint would not be violated at 5.7°C. The thermal plume temperature decreases quickly downstream of the GWHPs, such that there is minimal interaction between the GWHPs, and the maximum temperature in the domain is approximately 19°C. However, if the mass flow rate at each GWHP is increased from the default value of $\dot{m} = 0.1\ell/s$, the thermal plumes could be much larger and cause more thermal interaction.

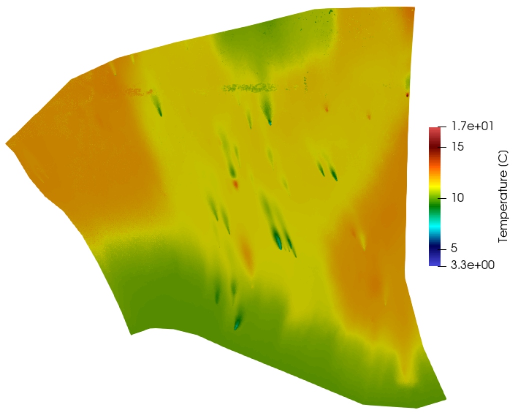
The groundwater temperatures for the Single-Region-Optimisation simulation-optimisation test case is shown in Figure 7.12, where the change in groundwater temperatures throughout the year can be observed. All sub-figures are plotted between 3.3°C and 17°C. The simulation begins on January 1^{st}. At day 60, some GWHPs are operating in cooling mode (which warms up the groundwater), but the impact of the GWHPs operating in heating mode is still observed with the minimum of 3.3°C. This is below the minimum injection temperature allowed according to Section 6.1.1, which is only allowed for existing-original GWHPs. Over the spring months (day 120 and day 180), the minimum groundwater temperature increases to a minimum of 4.9°C and 6.6°C, respectively. This indicates that fewer GWHPS are operating in heating mode as the air temperature warms up. At day 180 and day 240, a at least three large GWHPs operating in cooling mode appear, raising the maximum groundwater temperature to 17°C, still within the constraint limits.

Continuing to day 300 and day 360, the minimum groundwater temperature begins to reduce again (dark blue spots appear and dark red spots fall away) due to the cold winter months as GWHPs begin operating in heating mode again. Visualising the results helps to validate the correct operation of the GWHPs (whether they are operating in heating or cooling mode), and check that the maximum and minimum injection temperature constraints are always met.

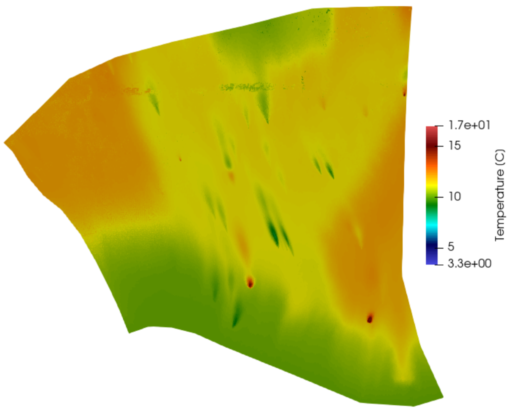
The selection of GWHPs for all test scenarios, except the trivial Baseline example, is shown in Table 7.6. All four test scenarios begin with 31 existing-original and 800 hypothetical GWHPs. Each scenario was run for 24 outer iterations. This was chosen to limit the build-up of too many GWHPs within one PFLOTRAN region, potentially impacting further downstream regions. Running the simulation-optimisation for 24 outer iterations allows for a total of 312 hypothetical GWHPs to be evaluated, with each of the 26 regions having 12 opportunities to add a new GWHP to the optimised solution. The final cost-optimised solution consists of 93 existing and 11 removed GWHPs for the Single-Region-Optimisation. This is substantially fewer than the potential 312 GWHPs. Examining Table 7.6, under the "Single-Region-Optimisation" columns, regions 4 and 28 having zero existing and removed GWHPs. This means that the optimiser never selects any GWHP to meet the required heating and cooling demand, and other cost effective alternatives exist. In both regions, the investment cost of the GWHP is much larger compared to other regions.



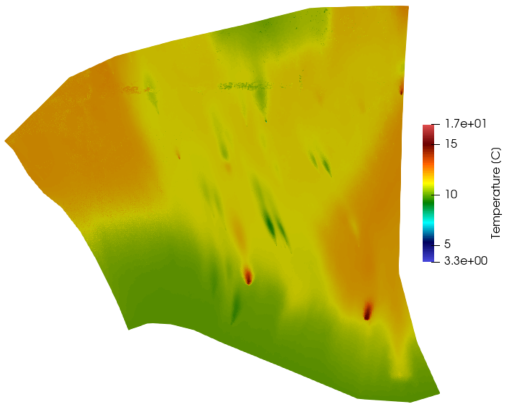
(A) Day: 60



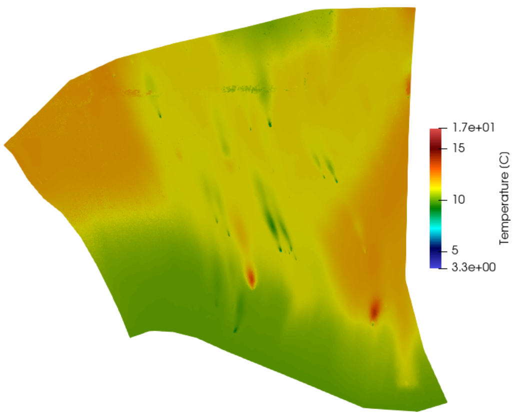
(B) Day: 120



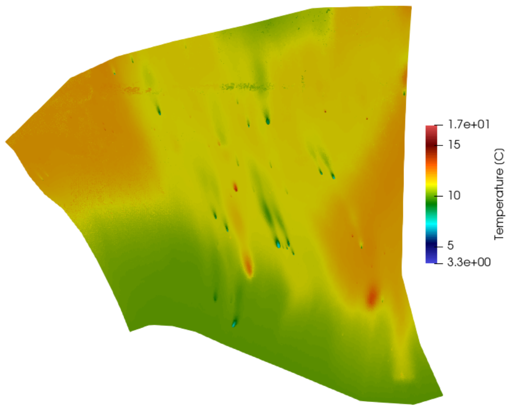
(C) Day: 180



(D) Day: 240



(E) Day: 300



(F) Day: 360

FIGURE 7.12 REG-30: groundwater temperatures for the Single-Region-Optimisation scenario at day 60, 120, 180, 240, 300 and 360. The temperatures are displayed on a slice at 525*m* above sea level.

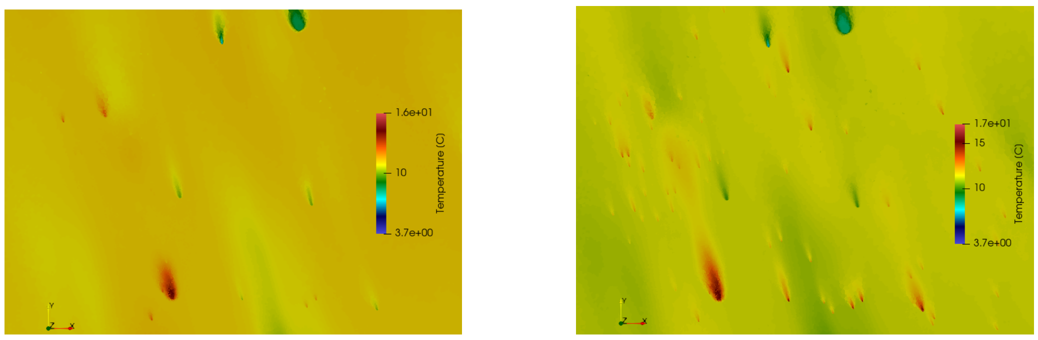
TABLE 7.6 REG-30: the number of existing ("Exis."), hypothetical ("Hyp.") and removed ("Rem.") GWHPs at the start of the simulation-optimisation coupling, and at the end of the Single-Region-Optimisation, Double-Region-Optimisation, Non-Coupled, and Zero-Cost test scenarios. The region number is displayed in the left column.

	Start		Single-Reg-Opt.		Double-Reg-Opt.		Non-Coupled		Zero Cost	
Region	Exis.	Hyp.	Exis.	Rem.	Exis.	Rem.	Exis.	Rem.	Exis.	Rem.
0	0	11	7	0	7	0	7	0	11	0
1	1	22	13	0	15	0	15	0	13	0
2	4	25	12	0	12	0	12	0	15	0
3	0	15	4	0	4	0	4	0	12	0
4	0	22	0	0	0	0	0	0	12	0
5	0	22	0	2	2	0	2	0	0	11
6	2	25	3	0	3	0	3	0	13	0
7	1	26	2	0	2	0	2	0	13	0
8	5	38	5	0	5	0	5	0	5	12
9	0	44	0	2	0	2	2	0	0	11
10	2	37	3	0	2	1	3	0	14	0
11	1	39	1	0	1	0	1	0	1	11
15	2	67	6	0	6	0	6	0	14	0
16	1	14	4	0	4	0	4	0	12	0
17	2	30	4	1	5	0	5	0	14	0
18	2	29	2	5	7	0	7	0	13	0
19	2	29	4	1	5	0	5	0	14	0
20	3	21	4	0	4	0	4	0	14	0
21	1	47	6	0	6	0	6	0	12	0
22	0	48	2	0	2	0	2	0	12	0
23	0	26	5	0	5	0	5	0	11	0
25	0	25	0	0	0	0	0	0	12	0
26	1	25	2	0	2	0	2	0	12	0
27	1	27	3	0	3	0	3	0	13	0
28	0	44	0	0	0	0	0	0	11	0
29	0	42	1	0	1	0	1	0	12	0
Total	31	800	93	11	103	3	106	0	285	45

Region 0 and region 1 have 7 and 13 existing GWHPs, respectively, and no removed GWHPs. Therefore, region 1 always added a new GWHP in every outer iteration, but region 0 eventually stops selecting more hypothetical GWHPs for the optimal solution, even though 4 hypothetical GWHPs still remain available.

The Double-Region-Optimisation also performs 24 outer iterations for each of the 26 urbs regions, where each region can potentially add a new GWHP to the optimised solution, theoretically allowing for a total of 624 new GWHPs. This comes at the expense of potentially removing more GWHPs due to constraint violations. The cost-optimal solution consists of 103 existing GWHPs and 3 removed GWHPs. Regions 0, 4 and 28 had the same results as in the Single-Region-Optimisation case, whereas region 1 has a total of 15 existing GWHPs. A big difference is observed for region 18, where no GWHPS were removed in the Double-Region-Optimisation case.

The Non-Coupled case allows for each region to simultaneously add as many GWHPs to the



(A) With Cost (B) Zero-Cost

FIGURE 7.13 REG-30: groundwater temperatures for with costs (left) and zero costs (right). The temperatures are displayed on a slice at 525*m* above sea level at day 360.

list of existing GWHPs in one outer iteration, without considering any external constraints. Hence, there are no GWHPs removed. This is obviously not an optimal solution, nor does it account for interaction between GWHPs, but is performed purely as a check for whether GWHPs are economically viable within the urbs regions. The final solution consisted of 106 existing GWHPs. This test verified that the lack of additional GWHPs in the Single-Region-Optimisation case is due to the optimiser no longer selecting new GWHPs after a few outer iterations. By altering the optimisation objective function, e.g., to minimise CO_2, or making the GWHPs cheaper to install or operate, the simulation-optimisation coupling may favour installing more GWHPs in the domain.

The lack of GWHPs installed or removed from the domain indicates that the GWHPs are not a cost effective solution. As a final test to ensure that the optimiser and software adapters were functioning correctly the investment cost, variable cost and fixed cost of each GWHP in each region was set to zero. Therefore, the cost-optimal objective function would favour installing GWHPs in the city as they are free to install and free to use. A final solution consists of 285 existing and 45 removed GWHPs after 24 outer iterations. An area in the middle of the REG-30 domain for the Zero-Cost and the Single-Reg-Optimisation scenario is shown in Figure 7.13. The increase in the number of GWHPs within the domain is clearly noticeable for the Zero-Cost scenario but does not significantly change the temperature within the domain. As the GWHPs are free to use, the amount that each GWHP is utilised also increases, resulting in a slightly higher maximum temperature. This final test proves that the current software adapters are functioning correctly, that the optimisation result is highly dependent on the urbs model, and that currently the GWHPs are not the cheapest options to meet the heating and cooling demands within the urbs model.

The current evaluation does not consider the quality of the optimal solution in terms of the use of various conversion processes. In future work, evaluating the amount of each conversion process (GWHPs, oil and gas heating, solar etc.) and how this varies between the various regions will be performed. Evaluating various scenarios is now also possible in future, such as adding a carbon cost by increasing the variable and fuel cost for oil and gas heating or implementing a subsidy for GWHPs by lowering the investment cost.

7.3 Summary of Chapter 7

The purpose of Chapter 7 was to present the simulation coupling results for two test cases with a high level of model complexity but varying model size. Firstly, the COM-4 model was introduced, which contains multiple complex features such as existing-original GWHPs with multiple injection and extraction wells, hypothetical GWHPs and subsurface culverts. The model was small enough to manually validate the nearest-neighbor data mapping procedure for the combined spatial-temporal coupling interface vertices, to test the performance of the quasi-Newton equation coupling method, and to ensure that the constraint checking procedure limited the GWHPs to operate within the pre-defined range. All these features were tested along with the staggered coupling approach developed in this thesis.

Secondly, the REG-30 model was introduced, which covers a large area of the city of Munich with 800 hypothetical GWHPs. The scalability performance of PFLOTRAN and urbs was studied. The VTK file output for visualisation was found to be the bottleneck for highly scalable PFLOTRAN simulations. Using a weekly time resolution for the energy infrastructure optimisation was found to be much faster than a daily time resolution, while still ensuring a sufficient accuracy. The simulation-optimisation test cases showed that most of the 800 hypothetical GWHPs were not selected for the cost-optimal solution, and that alternative means of meeting the heating and cooling demands were more cost-effective for the specific urbs models. This was further validated with the Non-coupled and Zero-Cost scenario. The REG-30 model proved that the developed coupling procedure and software adapters are capable of performing large-scale simulation-optimisation coupling for city-wide GWHP optimisation. The simulation-optimisation tool allows for regulatory bodies to perform high-fidelity simulations of the subsurface and help governing authorities to plan the roll-out of further GWHPs and evaluate the cost of various pricing policies.

8 Deep Learning for Shallow Subsurface Modelling

The ability to optimise the layout and usage of groundwater heat pumps (GWHP) using numerical simulations, as well as performing online monitoring and evaluation of GWHPs throughout a city, can help society move into a smarter, digital era. Creating virtual models of the city's subsurface allows authorities to constantly monitor the state of the subsurface aquifer and to locally optimise the placement and usage of GWHPs for individual applications. Current high-fidelity numerical simulations, as performed in Chapter 7, show that it is possible to optimise the problem of which GWHPs to use on a city-wide scale, but unfortunately, come at a high computational cost. Each virtual model requires generating a mesh of the subsurface domain, calibrating the subsurface model and creating an energy infrastructure model. Afterwards, each high-fidelity simulation requires large computing resources to solve the solution in a reasonable amount of time. This limits the ability to quickly perform numerical simulations for even a small domain around a single GWHP.

This is the perfect scenario where a cheaper digital twin can assist. A digital twin is a digital copy of a real-world asset that interacts with the physical world via sensors and measurements and can be used to test and optimise the physical asset virtually, which could then be used to predict the outcome of future scenarios [Gri17]; [Gla]; [Söd17]. By creating a digital twin that uses a fast and accurate surrogate model of a GWHP, real-time evaluation and inspection can be performed for GWHP performance modelling.

In this Chapter, a novel deep learning based model is developed that can quickly and accurately predict the thermal field around an active GWHP. The current standard analytical models that provide fast temperature plume prediction, as discussed in Section 8.3, are insufficient for complex groundwater scenarios. Section 8.1 provides examples of how and where deep learning surrogate models have been applied to physics simulations. An explanation of how a deep learning surrogate model can improve the current optimisation methodology is described in Section

8.2. The novel deep learning model developed in this work, along with the variations tested, is introduced in Section 8.4, followed by numerical testing and analysis of the results in Section 8.6.

8.1 Background

Surrogate modelling of complex and computationally expensive physics models is becoming common place in many applications. Surrogate models or low-fidelity models replace the computationally expensive high-fidelity model by a cheaper substitute. This is usually at the expense of the generalisability and accuracy of the model, while achieving a significant reduction in the computational expense. Therefore, surrogate modelling is a highly popular technique in the engineering field, where thousands of simulation evaluations are required to either optimise a design or perform uncertainty quantification with many input parameters and where some loss in accuracy can be tolerated. Applications benefiting from surrogate modelling range from reduced order models for turbulent flow [Hij20]; [Lin16]; [Dur19], optimisation of wind turbine layout [Kam16] and modelling reactive transport for power generation [Lau17]; [du 18] to geochemical simulations [Jat16].

For subsurface geothermal applications, surrogate modelling has been applied to numerous groundwater problems. Ansari [Ans14] developed a reduced-order model (ROM) using a trajectory piecewise linearisation approximation, which was improved upon by Trehan [Tre16] by using a trajectory piecewise quadratic approximations and proper orthogonal decomposition to model subsurface production reservoirs. Projection based methods have been a popular approach for creating ROMs for subsurface flows [Car09]; [Pas11]; [Li13]; [Boy14], which utilise the method of snapshots, i.e., using input-output pairs to generate the surrogate model. Kani and Elsheikh [Kan19] extended the POD based ROM with a deep residual recurrent neural network for subsurface multi-phase flow problems to create a "physics-aware" recurrent neural network. Most POD based methods still require access to the underlying physics solver to reduce the cost of the solving the linear system of equations and are not truly black-box surrogate models. Asher et. al. [Ash15] provide a thorough review on methods used for surrogate modelling of groundwater flow, including data-driven methods, projection based methods and multi-fidelity models.

A promising method of surrogate modelling lies in the field of deep learning. Deep learning utilises artificial neural networks (ANN), trained to quickly predict the solution to a problem given a set of input values. ANNs can be divided into various learning paradigms: supervised learning, unsupervised learning or reinforcement learning are the most common types of learning. Within supervised learning, the training data set contains the correct output data results for the corresponding input data. The network is trained to reduce the error between the network output and the correct output data.

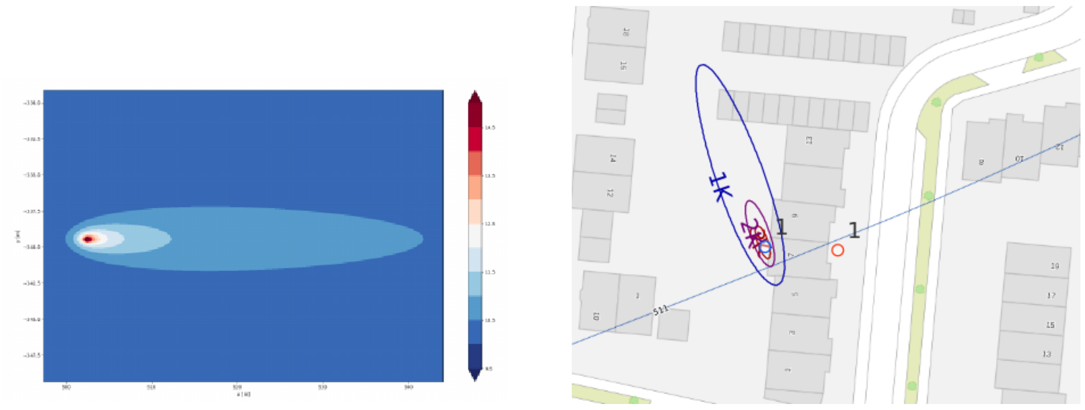
ANN's are quickly becoming popular in a variety of research areas, from reacting flows [Lau17], CO^2 storage [Pan14], predicting flow over aerofoils [Thu19], fluid-structure interaction [Gup22], to subsurface media [Wan20]; [Tan21]. Deep learning is not only being used to replace classical numerical simulations, but to also work alongside them. A popular deep learning method for computational mechanics is that of physics-informed neural networks (PINN) [Rai19]. A PINN uses the underlying partial differential equation to train the ANN instead of output data, as the solution (ANN output) must satisfy certain physical characteristics, such as conservation of mass. A PINN determines how much the network output violates the governing equations, e.g., including diffusion and advection terms and determines the respective output residual. This residual is used to penalise the ANN output in the network loss function to improve the solution over time. Laubscher [Lau21] implemented a PINN to predict the chemical composition of elements in reacting flows. Wang et. al. [Gao21] implemented a coarse to fine resolution solver, which used a convolutional neural network (CNN) and a PINN loss function to determine the fluid velocity at the finer resolution, where the solution had to satisfy the same physical laws as the coarse model input data. Kashefi et.al. [Kas21] developed a PINN to predict fluid flow, that accepted the geometry information as a point cloud and predicted the pressure and velocity field for a set of boundary conditions. For modelling groundwater problems, Zhu et. al. [Zhu19] used a PINN based autoencoder-decoder network to predict the groundwater pressure and Darcy velocity from the groundwater permeability field for a specified set of boundary conditions.

Deep learning is becoming a popular approach for developing surrogate models as they can be used in a simple black box fashion, where only training data are required to build a low-fidelity model. Therefore, complex models can be built from data alone even if the underlying physics are unknown. With the current computing power available to generate training data, deep learning models are everywhere.

8.2 Improving the Optimisation with Surrogates

The focus of the GEO.KW project is to optimise the layout and usage of GWHP's on a city-wide scale. As seen in Chapter 7, the numerical simulations of the subsurface were shown to be computationally expensive and require access to large computing resources. Therefore, implementing a cheap surrogate model will allow for the numerical evaluation of a GWHP with fewer computational resources. There are two cases where a surrogate could improve the current coupling methodology in the future.

- Firstly, the surrogate model can act as a complete replacement for the high-fidelity solver, evaluating each GWHP individually. If required, the local solutions can be combined into a global solution for the coupled model, with some error correction required.
- Secondly, the surrogate model can be used as a pre-processing step to detect significantly interacting GWHPs before the high-fidelity model run. This will reduce the probability of performing high-fidelity simulations for cases suggested by the optimiser where the GWHPs would almost certainly be interacting and the GWHP constraints would be violated, resulting in unnecessary high-fidelity model runs.



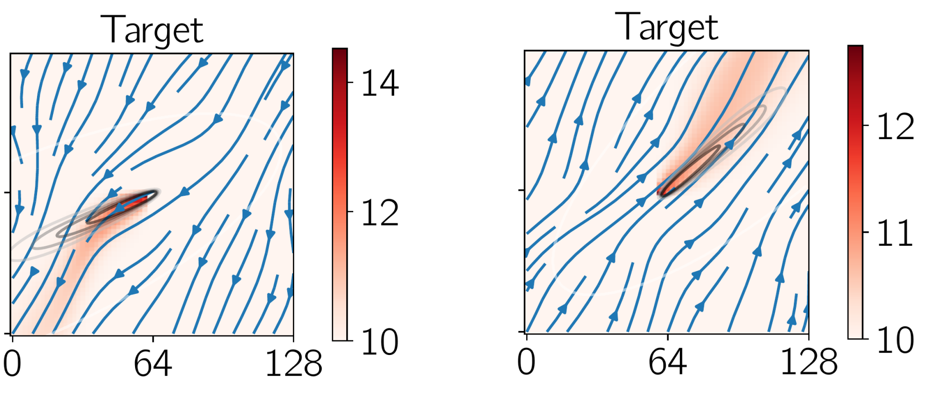
(A) LAHM Model (B) Online Evaluation Tool

FIGURE 8.1 Analytical temperature heat pump prediction with (A) thermal profile of the LAHM model and (B) LAHM model temperature isolines overlayed onto a street map view from an online evaluation tool developed within the GEO.KW project.

8.3 Analytical Surrogate

The commonality of both application cases in Section 8.2 is the requirement for an effective surrogate model. The current standard is to use analytical formulas, as they are widely used in practice. They offer a convenient and fast method for evaluating the local thermal impact a GWHP has on the surrounding subsurface [Pop20]. The LAHM model, described in Section 6.1.3, is a common analytical formulation and the current standard. However, the analytical formulation suffers from a few disadvantages: *(i)* it is uni-directional and does not account for changes in the Darcy flow direction, *(ii)* it has limited accuracy as it is dependent only on the local conditions at the GWHP itself, i.e., it assumes a constant permeability field. These limitations are not easy to resolve with an analytical formulation as the field information such as the permeability can be highly heterogeneous. An exemplary solution of the LAHM model in a 2D domain is shown in Figure 8.1 (left) and applied to an online evaluation tool (developed within the GEO.KW project) that overlays the analytical formulation on a city map, is shown in Figure 8.1 (right).

The disadvantage of the LAHM model uni-directionality is highlighted in Figure 8.2, where the LAHM solution is overlayed on top of the 2D numerical simulation with a heterogeneous permeability domain performed with PFLOTRAN. The uni-directional LAHM solution is not able to account for the change in Darcy flow direction and therefore cannot account for the change in thermal plume direction. A suitable surrogate model would need to be able to accommodate a reasonable amount of heterogeneity in the subsurface.



(A) Example A (B) Example B

FIGURE 8.2 Numerical and analytical solution for a 2D subsurface simulation. The analytical solution, represented by the isolines, is dependent on the velocity magnitude and direction at the location of the GWHP and does not account for the spatial change in flow direction. The simulated temperature plume follows the velocity streamlines as the solution is dominated by the advective term of Equation 5.4.

8.4 Framework for Heat Pump Modelling with Deep Learning

In this thesis, we propose to use an ANN to develop a surrogate model, that determines the subsurface aquifer temperature field after the addition of a GWHP. ANN models can learn highly complex solutions using only input-output data. However, this typically requires a large dataset to train the model. A surrogate model will only be useful if enough data can be obtained at a relatively low computational expense and be accurate enough for a variety of situations.

Developing a surrogate model involves two stages: the learning (offline) stage and the evaluation (online). To build a practically useful surrogate model, the online stage must be fast and computationally cheap. An expensive offline stage can be tolerated as it represents a once-off computationally expensive training period, including generating the training data.

8.4.1 Input data

A requirement for the input data of the surrogate model during the online stage is that they must use readily available information without requiring any high-fidelity numerical simulations. Therefore, the only information readily available is from the baseline simulations of each PFLOTRAN sub-domain. These baseline simulations contain the solution of the groundwater pressure and the Darcy velocity in the subsurface of the current state, or natural state, of the groundwater subsurface with only existing GWHPs installed. If the input data are derived from this baseline simulation, no additional numerical simulations need to be performed in each online evaluation to obtain the network input.

It is assumed that the thermal plume direction is dominated by the advection term (adv.) in the Darcy flow equation due to the thermal plume following the streamlines in Figure 8.2

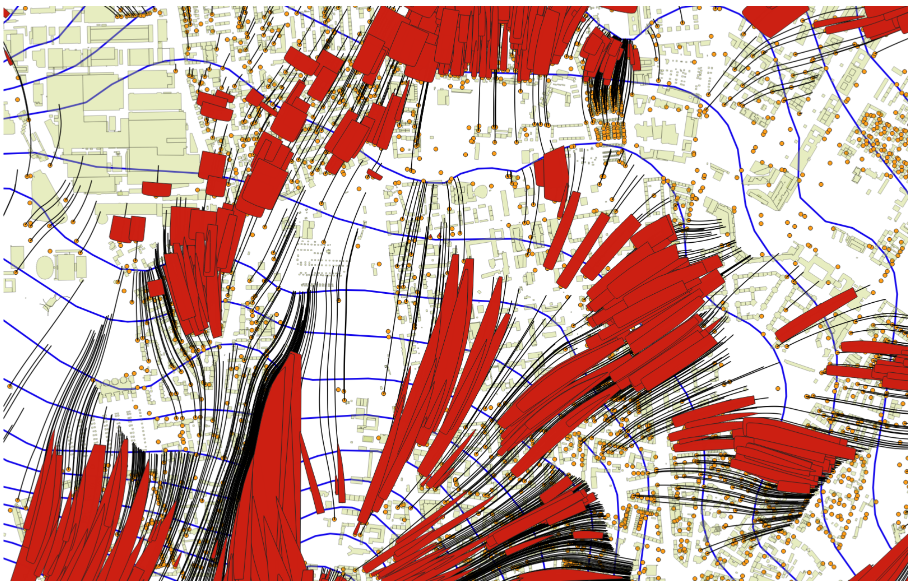


FIGURE 8.3 Streamlines of the Darcy velocity (black) with hypothetical thermal plumes (red) superimposed over the injection wells (orange dots). The streamlines are always perpendicular to the hydraulic pressure isolines (blue). The thermal plumes are generated by superimposing the LAHM analytical solutions (red), that have been morphed to follow the streamlines extending from the injection wells. The image was generated using QGIS (`https://www.qgis.org/en/site/`).

and not the diffusion term (dif.). Revisiting the subsurface model formulation in Section 5.3.1 and removing the time-dependent part of Equation 5.4, we find that the temperature profile is dependent on the Darcy velocity

$$\nabla \cdot (\underbrace{\eta \boldsymbol{q} H}_{adv.} - \underbrace{\kappa \nabla T}_{dif.}) = Q_e. \tag{8.1}$$

This is an over-simplification as the terms in Equation 5.2 are also temperature dependant, which would further influence the pressure field and subsequent Darcy velocities. However, we assume that the influence that the temperature has on the Darcy velocity is small, which strengthens the presumption that the already known Darcy velocities may hold relevant information to learn the temperature profile solution. As the Darcy flow information is already available from the baseline simulation, the network input data can be extracted for any location in the subsurface domain without performing any additional numerical simulation during the online stage in any real-world use case. The pressure isolines are shown in Figure 8.3 for a part of Munich, where the hydraulic pressure isolines (blue) can be used to determine the Darcy velocity streamlines (black) and velocity magnitude through $\boldsymbol{q} = K\Delta P$, where $\boldsymbol{q}$ is the velocity magnitude, K is the hydraulic permeability and ΔP is the pressure gradient.

8.4.2 Output data

The output data from the surrogate model must be accurate for a wide variety of inputs and orders of magnitude faster to attain than the high-fidelity model to remain useful. For this specific application, only the temperature field around a GWHP is required to build a useful surrogate model, the GWHP influence on the groundwater pressure is neglected. The network receives the Darcy velocity without the influence of the GWHP as the input and outputs the temperature field induced by the presence of a new active GWHP. Within this thesis, we only consider the steady-state thermal plume and do not consider the build-up of the plume over time. The temperature field information is not readily available from the baseline simulation of the velocity field and therefore needs to be generated before training can begin. As the purpose of the surrogate model is to avoid performing any numerical simulations as far as possible, the training phase can either be built on

1. a physics-informed neural network, i.e., supervised learning where the network output must satisfy a specified partial differential equation, in addition or instead of minimising the difference to known output generated from high-fidelity simulations,
2. a data-based neural network, i.e., the network output must match the known simulation results from already obtained numerical simulations.

In this work, we do not consider PINN's. However, generating the output data from the high-fidelity models of Chapter 7 would incur a large computational expense, too large even for the offline stage. Therefore, training the network must be performed with data obtained from small, computationally cheap, domains such that the temperature plume profiles for hundreds of velocity fields can be obtained from numerical simulations to train the surrogate model.

8.4.3 Network Architecture

The network architecture is extremely important, as it depends on and simultaneously influences the input and output data structure, as well as impacting the accuracy and speed of the network evaluation. The information from a 2D numerical simulation can be viewed as images (similar to Figure 8.1), where the value at each pixel relates to the Darcy velocity in the x-direction, y-direction and the temperature magnitude. Therefore, convolutional neural networks (CNN) are perfectly suited for the task which are often used for image based processing. CNNs have shown to be beneficial for learning physical behaviour from image-like data [Thu19]; [Zhu19]; [Gao21]; [Tan20]. A CNN is a deep learning network that applies a sliding kernel filter across a structured array of data and multiplying the kernel weights $\boldsymbol{W}_c$ with the underlying image. The kernel weights are the parameters that can be adjusted during the network training process. Many different kernel weights are applied to each image to extract different features. New kernels are applied to the already extracted features to further extract features at multiple scales, each time performed on the next layer of kernel weights. For brevity, we do not explain the derivation or

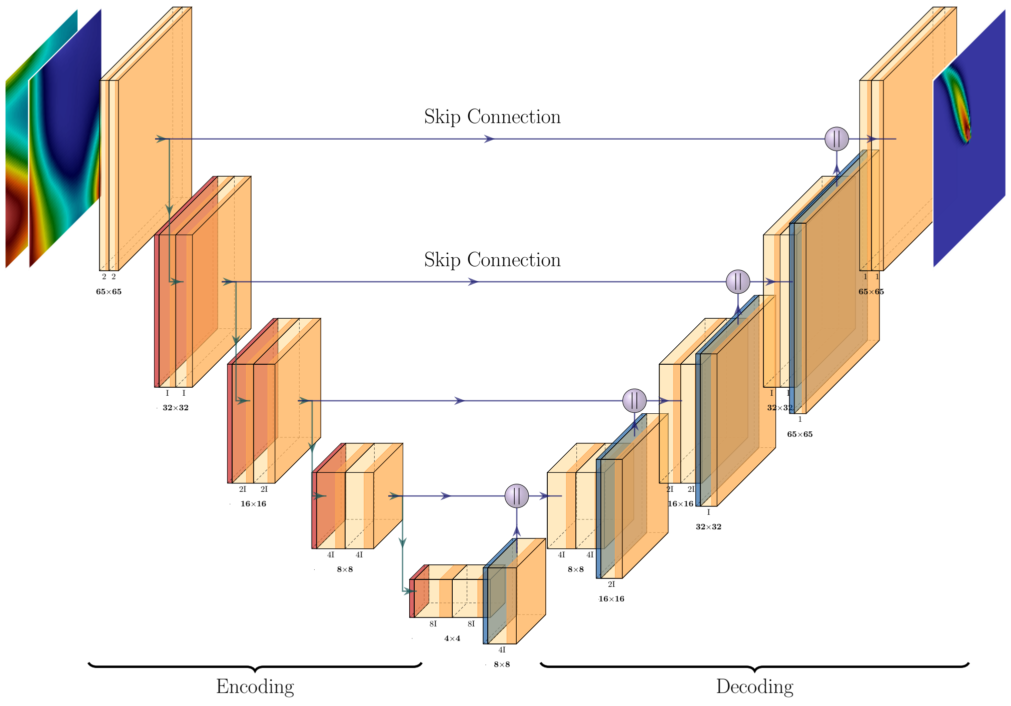


FIGURE 8.4 Initial TurbNetGeo-Light architecture for the convolutional neural network. The network contains a two channel Darcy velocity input with a single channel output of the temperature field. The TurbNetGeo network contains two more layers until the bottleneck contains only $32I\times1\times1$ (32 times the initial features). The TurbNetGeo-NoSkip-Light network does not have the skip connections from the encoding to decoding steps. The image is produced using the open-source visualisation tool from `https://github.com/HarisIqbal88/PlotNeuralNet`.

operation of CNNs here due to the abundance of literature on the subject [Agg18]; [Ska18] and practical examples[1].

In this thesis, three CNN variations based on the TurbNet architecture by Thuerey et. al. [Thu19], which is itself a variant of the U-Net architecture by Ronneberger et. al. [Ron15], are evaluated. The initial network architecture is depicted in Figure 8.4 with skip connections between the encoding and decoding steps. The skip connections help the decoding step to retain large-scale features that might get lost during the encoding step. The input data are provided as a two-channel image of the Darcy velocity in the x-direction $\varphi_{in,1}^{65\times65}$, and y-direction $\varphi_{in,2}^{65\times65}$ (q_x and q_y, respectively) of 65×65 pixels each, as depicted at the left of Figure 8.4. A 65×65 pixel image was selected such that the GWHP can be placed in the centre of the domain, having an equal number of surrounding pixels in both dimensions. The output $\varphi_{out}^{65\times65}$, is a single channel of the temperature profile throughout the domain.

Each encoding step is divided into multiple layers, each containing a rectified linear unit

[1]`https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks` (accessed on 18/10/2022).

(ReLU), max pooling of 2×2, stride of 2 for down-sampling the image size and batch normalisation. Each encoding step halves the image pixel size per direction and doubles the number of features. A feature map is produced by applying multiple filter kernels[2] over the input images. A larger number of initial features, i.e., features extracted in the first layer, may extract more information about the input images themselves. The total number of trainable parameters can be adjusted by varying the total number of layers and the initial features. The neural network architectures were developed together with Raphael Leiteritz from the department of Scientific Computing at the University of Stuttgart[3]. The three architectures tested in this thesis are:

1. TurbNetGeo: 6 layers with skip connections
2. TurbNetGeo-Light: 4 layers with skip connections
3. TurbNetGeo-NoSkip-Light: 4 layers without skip connections (TurbNetGeo-Light without the connections)

Therefore, the CNN is a mapping operation that maps the input images to an output image, using the designed network and trained kernel weights,

$$S\left(W_c, \varphi_{in,1}, \varphi_{in,2}\right) \mapsto \varphi_{pred}. \tag{8.2}$$

The purpose of the training procedure, using a supervised learning approach, is to minimise the difference between the known output and the network predictions,

$$\underset{W_c}{arg\,min} \sum_{i}^{N_{data}} \|\varphi_{pred} - \varphi_{out}\|. \tag{8.3}$$

TABLE 8.1 Number of trainable parameters for the convolutional neural network. The number of parameters increases when increasing the number of initial features extracted in the first convolutional layer.

Init. Feat.	TurbNetGeo	TurbNetGeo-Light	TurbNetGeo-NoSkip-Light
4	77,033	18,697	17,221
8	306,257	73,873	68,041
16	1,221,281	293,665	270,481
32	4,877,633	1,171,009	1,078,561

Having a larger number of trainable parameters in a network typically helps to learn more complex mappings between input and output data. However, this requires longer training times and more data. The total number of trainable parameters for multiple initial feature sizes is shown in Table 8.1.

[2]https://pytorch.org/docs/stable/generated/torch.nn.quantized.functional.conv2d.html?highlight=filter

[3]https://www.ipvs.uni-stuttgart.de/departments/sc/

8.4.4 Data Generation

A sufficient quantity of data is required to train the deep learning model. However, gathering simulation data on the scale of the simulations in Chapter 7 is computationally too expensive. Additionally, the network described above can only accept images of a limited size. Therefore, the model is restricted to an area around the GWHP only and the input/output data pairs can be generated for this smaller region. The data generation procedure is divided into three steps: *(i)* input field and boundary condition generation, *(ii)* without-heat-pump evaluation and *(ii)* with-heat-pump evaluation.

Input field generation: The purpose of the input field generation procedure is to create a wide variety of Darcy velocities and flow direction, such that the thermal plumes should extend in all directions with non-uniform flow profiles. This data set is used to train a network to predict the thermal plume for any Darcy velocity field. Therefore, randomised hydraulic permeability fields and pressure gradient boundary conditions are generated. Subsequently, a PFLOTRAN simulation is performed on each generated set to produce the Darcy velocity field without a GWHP and another to obtain the temperature field with the GWHP.

First, the domain and the mesh size are defined. The domain of 130m×130m×2m is divided into 65×65×1 grid cells (each grid cell is 2m×2m×2m). The GWHP is placed at the centre of the domain with equal number of grid points around the GWHP.

Secondly, the permeability field is generated by placing pilot points on a uniformly spaced grid throughout the domain (a 4×4 and 6×6 grid was used). Random permeability values were generated for each pilot point, varying between 1.13×10^{-7} and 3.77×10^{-11}. The random values are interpolated from the pilot points to the grid using radial basis function interpolation with thin-plate-splines basis function. This method was used to generate 800 random permeability fields.

Thirdly, a randomly generated pressure gradient boundary condition was applied in the x-direction and y-direction, varying in the interval $[-0.0006, 0.0006]$. The combination of pressure and permeability was found to provide reasonable Darcy velocity magnitudes.

Finally, combining the permeability fields and pressure boundary conditions resulted in 800 unique input datasets. The relatively small domain of 4225 finite volume elements on the 65×65×1 grid leads to a runtime of approximately 180s to generate 25 datasets, which is well within the computational cost margins for generating data in the offline stage.

Without-heat-pump evaluation: The input data required by the CNN are the Darcy flow velocities throughout the domain, $\boldsymbol{q}_x$ and $\boldsymbol{q}_y$, without the influence of the GWHP. This emulates the conditions of any input data that would be fed into the CNN derived from a domain from Chapter 7. Therefore, the GWHP injection mass flow rate was set to zero for all 800 input fields to obtain the Darcy flow velocity without the influence of the GWHP (input data).

With-heat-pump evaluation: The correct thermal profile for each input field generated must be used for training the CNN using supervised learning. Therefore, all simulations performed in the *Without-heat-pump* step are repeated, where the GWHP mass flow rate was set to $0.05\ell/s$ at 15°C, against a background temperature of 10°C. This was repeated for all 800 input fields to provide the temperature field output (output data) for supervised training.

Pre-processing: The accuracy of the CNN model can be greatly improved by suitable pre-processing of the data. Firstly, 10°C was subtracted from the temperature output, as this was the background temperature field and initial groundwater temperature. Secondly, each quantity (Darcy velocities and temperature) was normalised to the range $[-1, 1]$ over the whole set of all input and output data sets. Any output result from the network is then inversely scaled back to the original range. The background temperature of 10°C is added to the temperature result to obtain the final output.

8.5 Experimental Setup

8.5.1 Software

All numerical groundwater simulations were performed using PFLOTRAN v3.0[4] to generate the input and output datasets. The CNN was built using PyTorch[5]. The PyTorch networks are built in Python and provided along with the training and testing data in the DaRUS dataset "Replication Data for: Geothermal-ML – predicting thermal plume from groundwater heat pumps" [Dav22c].

8.5.2 Hardware

The networks were trained on an NVIDIA GeForce RTX 3090 GPU with 24Gb RAM at the University of Stuttgart, running Ubuntu 20.04.

8.5.3 Training Configuration

All CNN networks were trained using the Adam Optimiser [Kin14], with a fixed learning rate of 0.0005 and a batch size of 64. The networks were trained for 50,000 epochs which was assumed to be sufficient to achieve a reasonable accuracy, while the testing set was evaluated every 10,000 epochs. Of the 800 samples, 650 samples were used for training and 150 samples were used for testing. The 800 samples were divided into 32 groups of 25 samples each. Groups 5, 10, 15, 20, 25 and 30 were used for testing, ensuring that all networks were trained and tested with the same sample set. A data-driven loss function was used to ensure the network was sufficiently accurate

[4]`https://bitbucket.org/pflotran/pflotran/wiki/Home`
[5]`https://pytorch.org/`

without any PDE information. The mean squared error (MSE) loss function for the network was defined as the error between known solution and prediction

$$MSE = \frac{1}{N_{data}} \sum_{i}^{N_{data}} \left(\boldsymbol{T}^{i}_{Pred} - \boldsymbol{T}^{i}_{Target} \right)^2, \tag{8.4}$$

where $\boldsymbol{T}^{i}_{Pred}$ is the temperature prediction for dataset i, $\boldsymbol{T}^{i}_{Target}$ is the known solution and N_{data} is the total number of data sets used for either training ($N_{data} = 650$) or testing ($N_{data} = 150$).

8.6 Validation of the Deep Learning Model

Purpose: The CNN validation is divided into multiple steps. The first step is to evaluate the training and testing loss of each network. This shows how the loss values are influenced by the network architecture and the number of initial features. Secondly, the network test predictions are categorised into three groups: good, medium and bad. In consultation with experts[6], a surrogate model that has a prediction error less than 1°C for heterogeneous flow profiles is still somewhat useful and an error below 0.5°C is very useful. Therefore, any prediction with the maximum error < 0.5°C is defined as good, any prediction with 0.5°C < maximum error < 1°C is defined as medium, and all others are defined as bad. Finally, various network prediction outputs are plotted to observe how the network temperature output performs, i.e., does the largest error occur at the heat pump injection site or does the thermal plume error increase further down the plume.

Results: The training loss and testing loss for the three network architectures are shown in Table 8.2 at the 50,000th epoch.

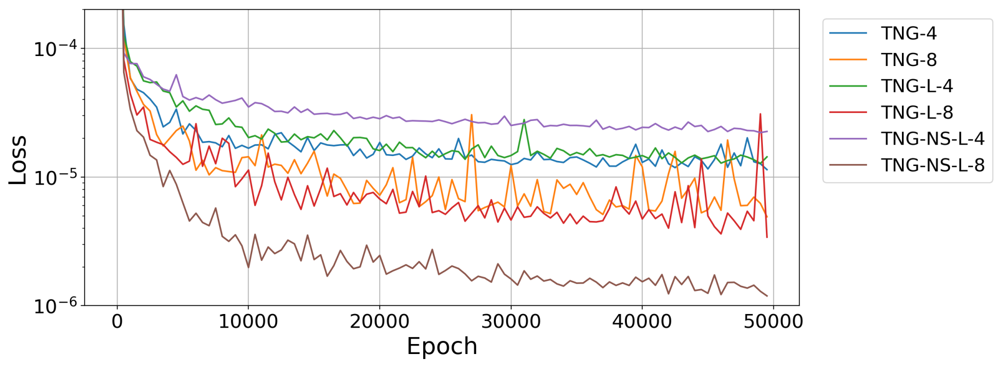
TABLE 8.2 Training and testing loss of all networks with varying number of initial features. The three networks were trained with 4, 8, 16 and 32 initial features ('Init. Feat.'), which varies the total number of training parameters in the network. The training loss ('Loss') and the testing loss ('Test Loss') are evaluated at the 50,000th epoch.

	TurbNetGeo		TurbNetGeo-Light		TurbNetGeo-NoSkip-Light	
Init. Feat.	Loss	Test Loss	Loss	Test Loss	Loss	Test Loss
4	$1.19 \cdot 10^{-5}$	0.0659	$1.38 \cdot 10^{-5}$	0.0556	$2.40 \cdot 10^{-5}$	0.0224
8	$6.22 \cdot 10^{-6}$	0.0455	$3.93 \cdot 10^{-6}$	0.0423	$1.26 \cdot 10^{-6}$	0.0173
16	$2.22 \cdot 10^{-7}$	0.0812	$3.31 \cdot 10^{-7}$	0.0318	$3.96 \cdot 10^{-7}$	0.0171
32	$2.16 \cdot 10^{-7}$	0.0433	$2.17 \cdot 10^{-7}$	0.0270	$2.66 \cdot 10^{-7}$	0.0159

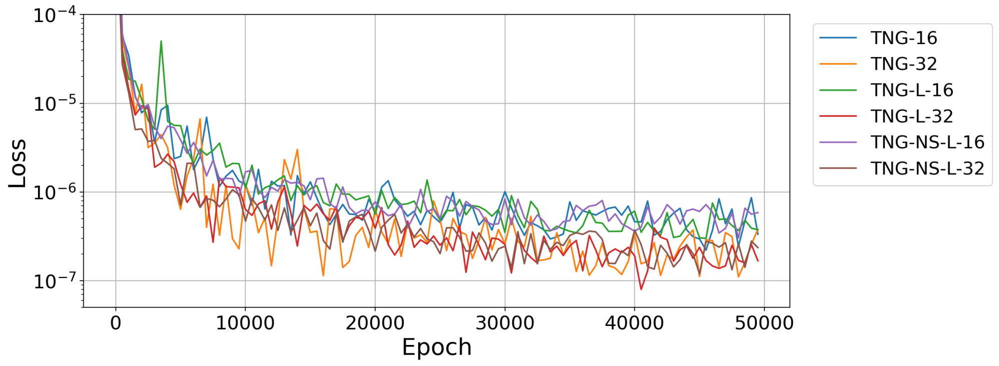
The training loss across all 50,000 epochs for all three network architectures with 4 and 8

[6]Chair of Hydrogeology, Technical University of Munich.

initial features are shown in Figure 8.5 (top) and with 16 and 32 initial features in Figure 8.5 (bottom). For each case, the loss value is only plotted for every 500th epoch, with no smoothing performed on the data. The testing loss, measured every 10,000 epochs for all three network architectures with 4 and 8 initial features, are shown in Figure 8.6 (top) and with 16 and 32 initial features in Figure 8.6 (bottom).



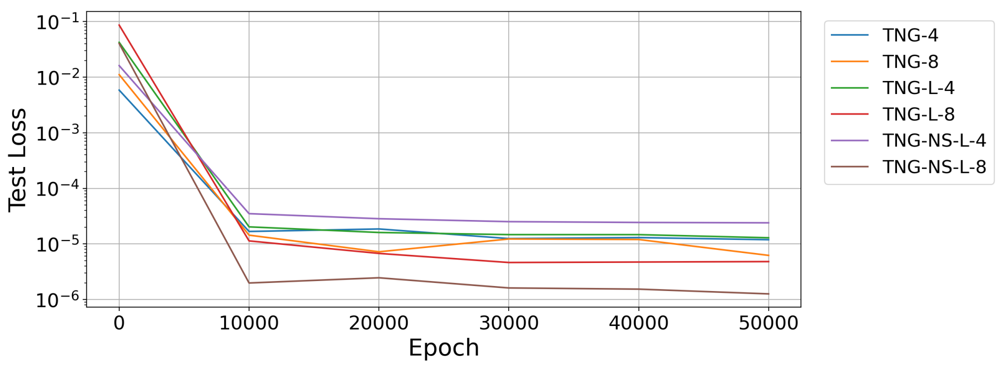
(A) 4 and 8 Initial Features



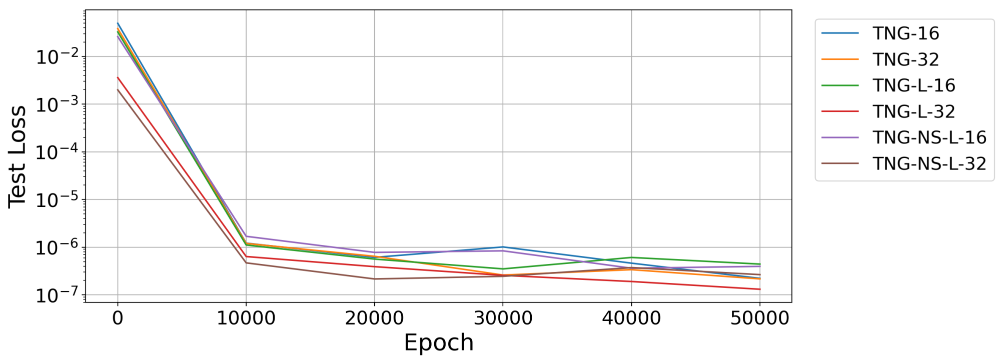
(B) 16 and 32 Initial Features

FIGURE 8.5 Mean square error training loss for all three network architectures for 50,000 epochs. The three network architectures TurbNetGeo ('TNG'), TurbNetGeo-Light ('TNG-L') and TurbNetGeo-NoSkip-Light ('TNG-NS-L'), were evaluated for 4 and 8 initial features (A) and 16 and 32 initial features (B), denoted as '-4', '-8', '-16' and '-32' in the legend, respectively. The training loss is plotted at every 500th epoch.

The total number of predictions categorised as good, medium and bad for each network and varying number of initial features, is shown in Table 8.3. For each network, a total of 150 test samples were evaluated and categorised according to the maximum error ϵ_{max} across all pixels: *(i)* good – $|\epsilon_{max}| < 0.5$, *(ii)* medium – $0.5 < |\epsilon_{max}| < 1.0$ and *(iii)* bad – $1.0 < |\epsilon_{max}|$.



(A) 4 and 8 Initial Features



(B) 16 and 32 Initial Features

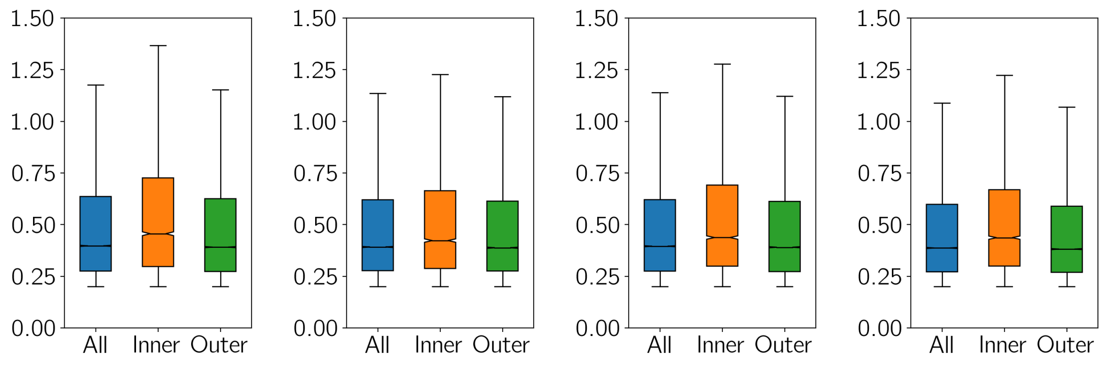
FIGURE 8.6 MSE testing loss for all three network architectures for 50,000 epochs. The three network architectures TurbNetGeo ('TNG'), TurbNetGeo-Light ('TNG-L') and TurbNetGeo-NoSkip-Light ('TNG-NS-L'), were evaluated for 4 and 8 initial features (A) and 16 and 32 initial features (B), denoted as '-4', '-8', '-16' and '-32' in the legend, respectively. The testing loss is plotted at every 10,000th epoch.

TABLE 8.3 Number of samples classified as 'good', 'medium' and 'bad' predictions for all three network architectures. Any prediction with the maximum absolute error $|\epsilon_{max}| < 0.5$°C is defined as 'good', any prediction with 0.5°C $< |\epsilon_{max}| <$ 1°C is defined as 'medium' and all others are defined as bad.

	TurbNetGeo			**TurbNetGeo-Light**			**TurbNetGeo-NoSkip-Light**		
Init. Feat.	Good	Medium	Bad	Good	Medium	Bad	Good	Medium	Bad
4	0	5	145	0	3	147	0	1	149
8	1	50	99	1	28	121	1	43	106
16	7	81	62	14	56	80	8	64	78
32	24	66	60	22	63	65	15	69	66

For the TurbNetGeo-Light (TNG-L) and TurbNetGeo-NoSkip-Light (TNG-NS-L) networks with 16 and 32 initial features, box plots (box and whisker plot) of the per-pixel error magnitude across all 150 test samples are plotted in Figure 8.7(A) – (D). Each box provides the median Q_2 (middle black line), lower quartile Q_1 and upper quartile Q_3 (box edges). The top and bottom lines (whiskers) define the 1.5× interquartile range[7] (IQR), i.e., 1.5·(Q_3 - Q_1). The upper whisker plots the maximum value that falls within the 1.5·IQR, and the bottom whisker plots the minimum value that falls within the 1.5 IQR.

For each test (A) to (D), the box plots were generated using values where the error was more than 0.2°C. We do not want to consider points far away from the plume, where the background temperature is still 10°C but a small error may be present in the prediction. Furthermore, all points within an inner region of 7 pixels from the heat pump injection site in each direction are shown in the 'Inner' plots and all other points are shown in the "Outer" plots. Examples of the prediction versus the target output is shown in Figure 8.8, where the majority of the pixels are in areas of little relevance.



(A) TNG-L-16 (B) TNG-L-32 (C) TNG-NS-L-16 (D) TNG-NS-L-16

FIGURE 8.7 Per-pixel error magnitude box plots for the TurbNetGeo-Light and TurbNetGeo-NoSkip-Light network architectures with 16 and 32 initial features for all 150 test samples. Each box provides the median Q_2, lower quartile Q_1 and upper quartile Q_3. The top and bottom edges (whiskers) define the 1.5 interquartile range (1.5·(Q_3 - Q_1)) values of the error magnitude. Only error values large than 0.2°C were added to the dataset for the box plots.

Detailed information for the box plots is provided in Table 8.4, including the maximum error (Max), upper whisker value from Figure 8.7 (Top IQR), upper inter-quartile Q_3, median (Med.), lower inter-quartile Q_1, minimum error (Min) and the percentage of pixels where the error is between 'Top IQR' and 'Max' (% Max), i.e., the number of pixels where the error is larger than the upper whisker value. The upper whisker value is determined as Top IQR $= Q_3 + 1.5 \cdot (Q_3 - Q_1)$.

Examining the training and testing loss conveys how well the network performs overall.

[7]`https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html?highlight=boxplot#matplotlib.pyplot.boxplot`

However, it does not indicate where or how the greatest errors occur, i.e., whether it occurs at the GWHP location itself or if the thermal plume is unable to follow the streamlines. Therefore, a qualitative assessment of the network's performance is required to gain further insight. Four different network predictions for the TurbNetGeo-Light network with 32 initial features from the 'good', 'medium' and 'bad' categories are shown in Figure 8.8, Figure 8.9 and Figure 8.10, respectively. Finally, the difference between four test sample predictions from the TurbNetGeo-Light (TNG-L) and the TurbNetGeo-NoSkip-Light (TNG-NS-L) with 32 initial features from the 'bad' category are provided in Figure 8.11 and Figure 8.12.

TABLE 8.4 Box plot data for the TurbNetGeo-Light and TurbNetGeo-NoSkip-Light networks with 32 initial features. The maximum error ('Max'), 1.5· inter-quartile range/top edge of box plot ('Top IQR'), upper inter-quartile ('Q_3'), median ('Med'), lower inter-quartile ('Q_1'), minimum value ('Min') and number of data points between 'Max' and 'Top IQR' ('% Max'). The maximum value for each row category is highlighted.

	TNG-L-16			**TNG-L-32**		
	All	Inner	Outer	All	Inner	Outer
Max [°C]	3.95	3.95	3.21	3.81	3.81	3.35
Top IQR [°C]	1.18	1.37	1.15	1.14	1.23	1.12
Q3 [°C]	0.64	0.73	0.63	0.62	0.66	0.61
Med. [°C]	0.40	0.46	0.39	0.39	0.42	0.39
Q1 [°C]	0.28	0.30	0.27	0.28	0.29	0.28
Min [°C]	0.20	0.20	0.20	0.20	0.20	0.20
% Max	5.76%	5.78%	5.69%	5.55%	5.11%	5.64%
	TNG-NS-L-16			**TNG-NS-L-32**		
	All	Inner	Outer	All	Inner	Outer
Max [°C]	3.73	3.73	3.07	3.62	3.62	3.53
Top IQR [°C]	1.14	1.28	1.12	1.09	1.22	1.07
Q3 [°C]	0.62	0.69	0.61	0.60	0.67	0.59
Med. [°C]	0.40	0.44	0.39	0.39	0.44	0.38
Q1 [°C]	0.28	0.30	0.27	0.27	0.30	0.27
Min [°C]	0.20	0.20	0.20	0.20	0.20	0.20
% Max	5.00%	5.35%	4.99%	4.39%	5.20%	4.43%

Discussion: First, we examine the difference between networks for the same number of initial features in Table 8.2. The training loss for all three networks with 4 and 8 initial features are at least an order of magnitude larger than 16 and 32 initial features. The TurbNetGeo-NoSkip-Light has the largest training loss at 50,000 epochs for both 16 and 32 initial features. The TurbNetGeo network has the lowest training loss for both 16 and 32 initial features, but only slightly and with 4 times the number of trainable parameters than the other two networks. Despite having fewer trainable parameters than the other networks, the TurbNetGeo-NoSkip-Light has the lowest testing loss. Overall, all three network architectures show similar training losses and testing losses for the same number of initial features.

Examining the training loss over time in Figure 8.5, a noticeable difference is observed between networks for 4 and 8 initial features. However, there is little difference between the networks with an equivalent number of 16 or 32 initial features. It is clear that the number of initial features influences the training loss more than the number of network layers. The testing loss is examined in Figure 8.6. Minor, if any, improvement in the testing loss is observed after the first test at 10,000 epochs, indicating that additional training does not aid in improving the real-world capability of the network on predicting the thermal plume on unseen Darcy velocity data. This would perhaps require more training and testing data to improve, or a fundamental change to the network is required to reduce the testing loss.

The categorisation of the prediction for each network and number of initial features into 'good', 'medium' and 'bad', is shown in Table 8.3. The TurbNetGeo network performs only moderately better than TurbNetGeo-Light, but at a higher training cost. We therefore focus on the TurbNetGeo-Light and TurbNetGeo-NoSkip-Light networks as the preferred networks for further analysis. The TurbNetGeo-Light has more test samples categorised as good for 16 and 32 initial features and more samples categorised as medium for 16 initial features. Even though the testing loss is lower for the TurbNetGeo-NoSkip-Light network, there may be few pixels where the error is large, forcing test samples to be categorised as 'medium' or 'bad'. However, we select the TurbNetGeo-Light with 32 initial features as the current default network for the rest of the analysis.

Table 8.3 only provides information on the maximum error of the prediction but does not provide information on the range of error magnitudes in the prediction. Therefore, the box plot in Figure 8.7 plots the median, interquartile range and maximum and minimum values (excluding outliers) of the per-pixel errors for the TurbNetGeo-Light and TurbNetGeo-NoSkip-Light networks across all 150 samples. The error at each pixel was determined and only included in the box plot data set if the error was larger than 0.2°C in order to ignore small errors in the prediction for regions far away from the plume, i.e., the box plot only include error values close to or within the thermal plume. Separate box plots were generated for all error values within 7 pixels from the heat pump (Inner) and all other pixels (Outer). This helps identify if the errors are, in general, larger near the heat pump or further away.

Across all box plots, the median error is below 0.46°C, indicating that at least half of the pixels where the error is above 0.2°C are also below 0.5°C. The largest Q_3 occurs for the TNG-L-16 'Inner' with a magnitude of 0.73°C, meaning 75% of error values are beneath this value. As the 'bad' predictions were included in the data set, this shows that the networks are able to provide reasonably good predictions and only a small number of pixels have a large error. The 'Inner' box plots have larger values than the 'All' plots as the error tends to be slightly larger in the middle of the domain, leaving fewer small error values to pull the box plot down. The upper most edge 1.5·IQR (whisker) is determined by $1.5{\cdot}\text{IQR} = Q_3 + 1.5 \cdot (Q_3 - Q_1)$ and finding the largest value that fits within this range. From Table 8.4, a maximum of 5.78% of all data points are above the 1.5·IQR value (% Max for TNG-L-16). Therefore, approximately 94% of all per-pixel errors are below 1.37°C, far from the maximum error of 3.95°C and close to the medium-to-bad criterion of

1°C, which only occurs for 1 pixel on 1 test sample for TNG-L-16.

Four test samples from the ’good’ category for the TurbNetGeo-Light with 32 initial features are shown in Figure 8.8. The maximum error occurs at the heat pump location for all four samples. However, the second sample has very small errors in the thermal plume in comparison to the others. The top and bottom samples indicate a good ability to follow the streamline as its path changes. The similar looking middle samples show how the thermal plume can spread out when the streamlines diverge (middle top), whereas the thermal plume remains narrow when the streamlines remain straight (middle bottom). As expected, the ’good’ category performs well.

Four test samples from the ’medium’ category for the TurbNetGeo-Light with 32 initial features are shown in Figure 8.9. The top two samples have a very low error magnitude in the thermal plume and was categorised into ’medium’ due to a small error spike at the heat pump location. The large error occurs when the GWHP location in the plume prediction is shifted one or two pixels compared to the target solution and the maximum temperature pixel is shifted. If shifted in the wrong direction, i.e., if the plume extends downwards but the GWHP pixel location is shifted upwards, a large difference exists at this shifted location only and artificially inflates the maximum error value. However, the error values within the thermal plumes themselves are well within acceptable limits. The bottom two samples show that the plume can morph with the streamlines, but not well enough to keep the error below 0.5°C.

Four test samples from the ’bad’ category for the TurbNetGeo-Light with 32 initial features are shown in Figure 8.10. Each sample indicates various problems with some predictions. The top sample cannot accurately capture the temperature far downstream, whereas the second sample does not follow the streamline, but cuts across it. The third sample cannot capture the high temperature at the heat pump location, which is also wide due to having a low Darcy velocity at the heat pump. Finally, the bottom sample is unable to capture the thermal profile near the heat pump.

Four test samples from the ’bad’ category for the TurbNetGeo-NoSkip-Light (first and third image) and TurbNetGeo-Light (second and fourth image) with 32 initial features, comparing the ability of the two networks, are shown in Figure 8.11 and Figure 8.12, each will two samples. For the first sample in Figure 8.11 (sample 20), the TNG-NS-L network provides a better prediction than the TNG-L network, which cannot predict the entire plume correctly. The largest error occurs further downstream of the GWHP. However, the TNG-L network provides a better prediction of the downstream plume compared to the TNG-NS-L for sample 51.

In Figure 8.12, the prediction for the first sample is similar for both networks, where the largest error is close to the GWHP. However, the TNG-NS-L network has a wider plume and does not quite follow the streamline as well as the TNG-L network. The final prediction is difficult for both networks. Both fail to predict the maximum temperature directly downstream of the GWHP while simultaneously over-predict the width of the plume.

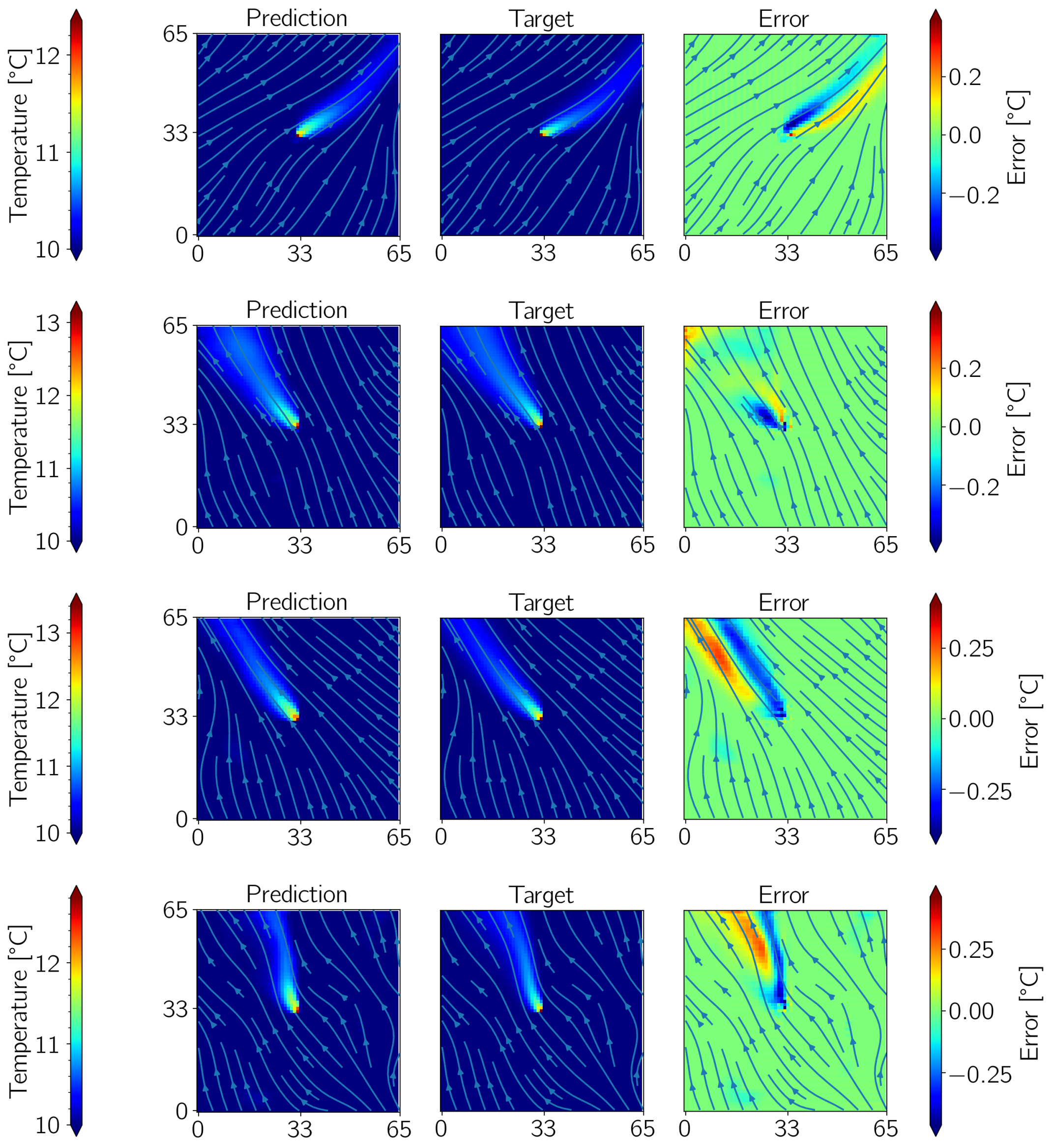


FIGURE 8.8 TurbNetGeo-Light: 'good' network prediction for four samples with 32 initial features. The network prediction (left) and the known target solution (middle) is plotted with the absolute Temperature [°C] (left bar). The error (right) between the prediction and target is plotted with the Error [°C] (right bar). The test sample numbers listed from top to bottom: 35, 96, 100 and 136.

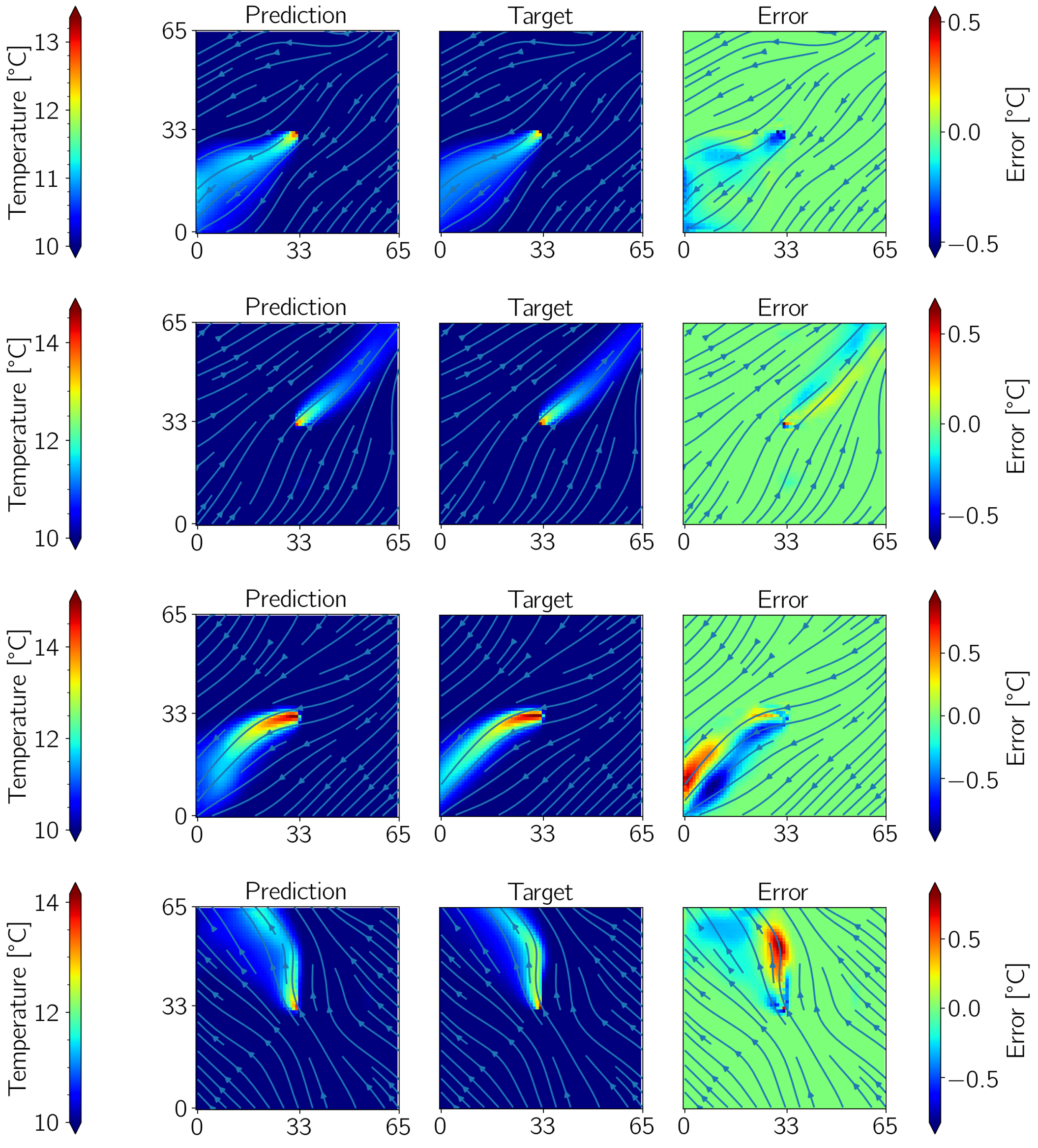


FIGURE 8.9 TurbNetGeo-Light: 'medium' network prediction for four samples with 32 initial features. The network prediction (left) and the known target solution (middle) is plotted with the absolute Temperature [°C] (left bar). The error (right) between the prediction and target is plotted with the Error [°C] (right bar). The test sample numbers listed from top to bottom: 16, 76, 99 and 12.

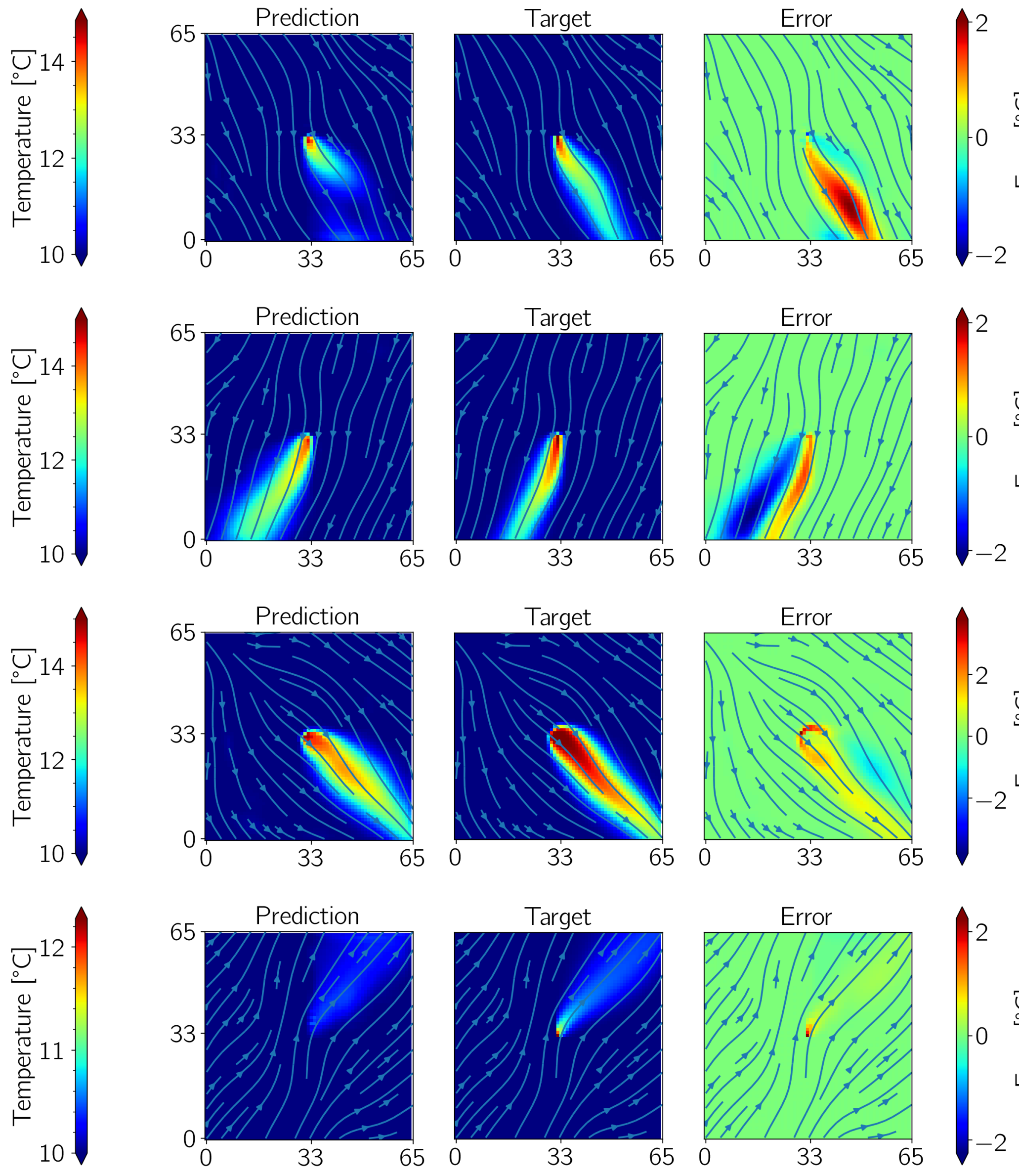
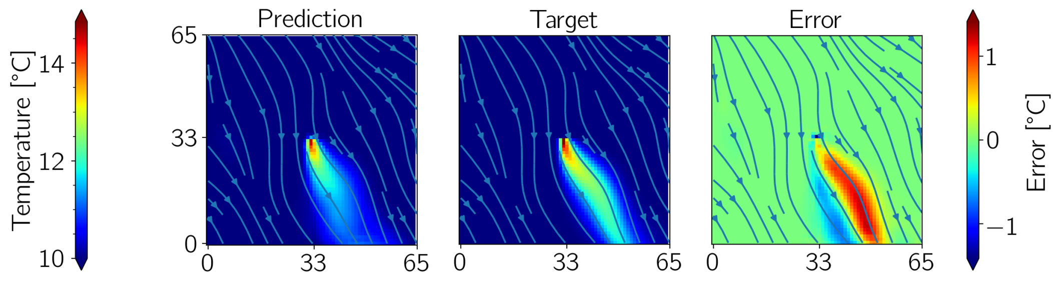
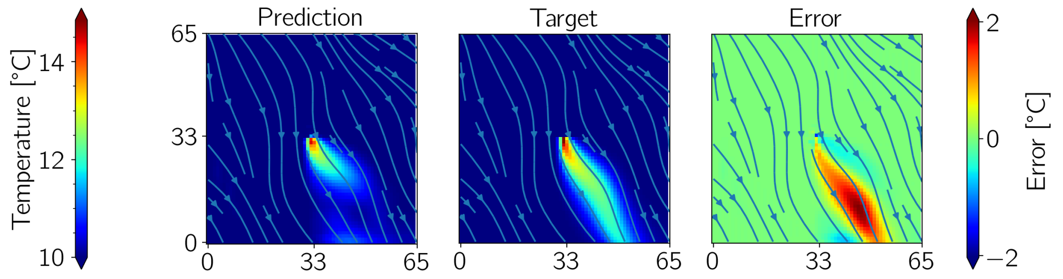


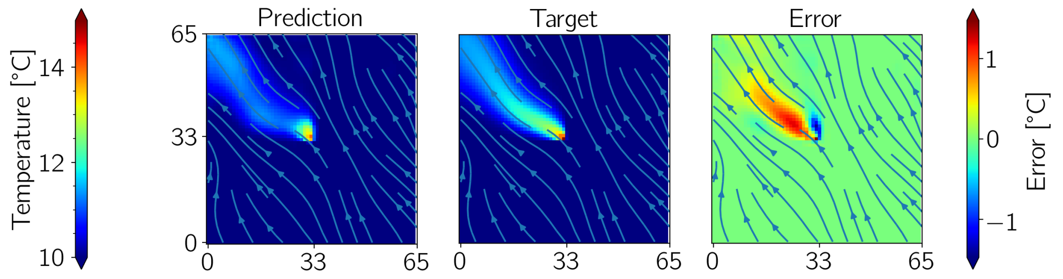
FIGURE 8.10 TurbNetGeo-Light: 'bad' network prediction for four samples with 32 initial features. The network prediction (left) and the known target solution (middle) is plotted with the absolute Temperature [°C] (left bar). The error (right) between the prediction and target is plotted with the Error [°C] (right bar). The test sample numbers listed from top to bottom: 20, 52, 66 and 139.



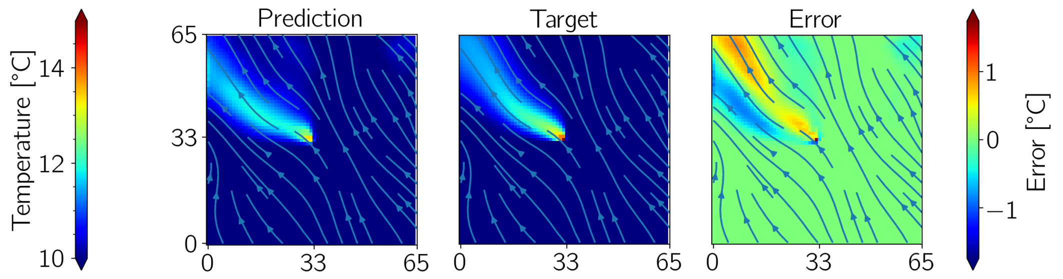
(A) TNG-NS-L: sample 20



(B) TNG-L: sample 20

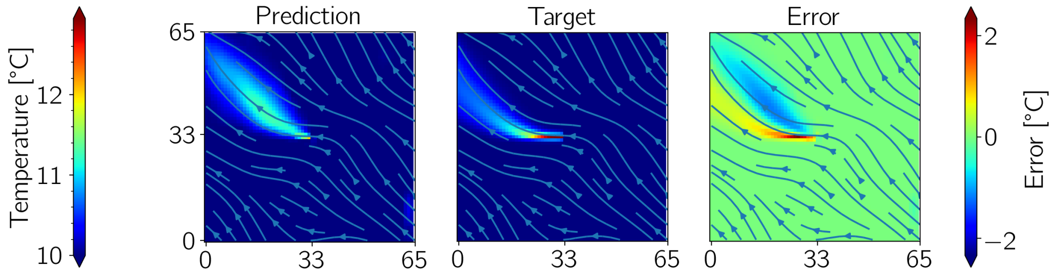


(C) TNG-NS-L: sample 51

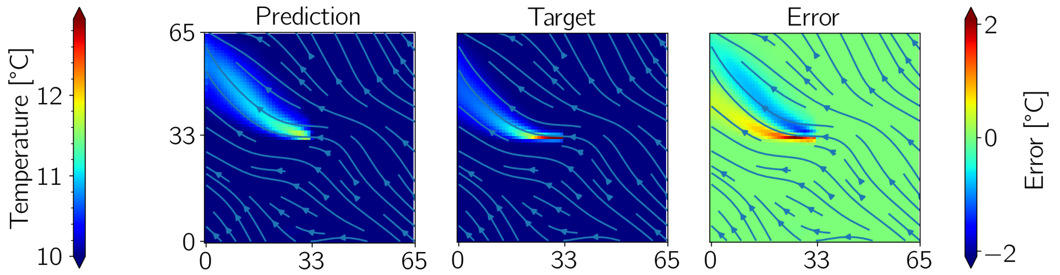


(D) TNG-L: sample 51

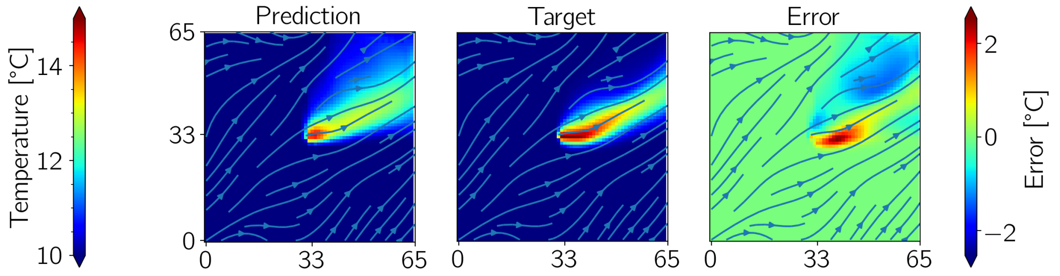
FIGURE 8.11 TurbNetGeo-Light versus TurbNetGeo-NoSkip-Light: 'bad' network prediction for two samples with 32 initial features. The network predictions from top to bottom are: TNG-NS-L – 20, TNG-L – 20, TNG-NS-L – 51, TNG-L – 51. The network prediction (left) and the known target solution (middle) is plotted with the absolute Temperature [°C] (left bar). The error (right) between the prediction and target is plotted with the Error [°C] (right bar).



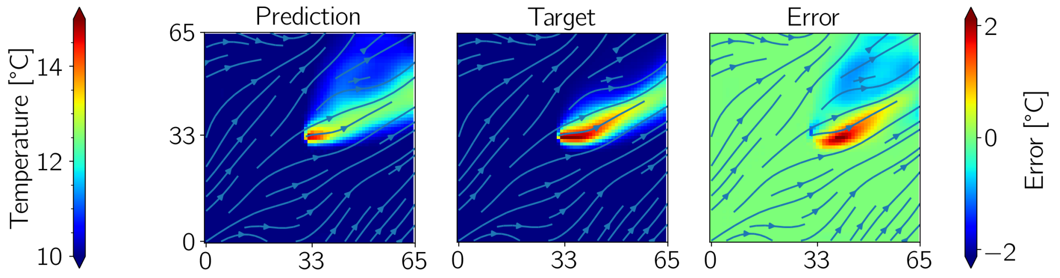
(A) TNG-NS-L: sample 137



(B) TNG-L: sample 137



(C) TNG-NS-L: sample 50



(D) TNG-L: sample 50

FIGURE 8.12 TurbNetGeo-Light versus TurbNetGeo-NoSkip-Light: 'bad' network prediction for two samples with 32 initial features. The network predictions from top to bottom are: TNG-NS-L – 137, TNG-L – 137, TNG-NS-L – 50, TNG-L – 50. The network prediction (left) and the known target solution (middle) is plotted with the absolute Temperature [°C] (left bar). The error (right) between the prediction and target is plotted with the Error [°C] (right bar).

By comparing the two networks, both have their own positives and negatives. The TurbNetGeo-NoSkip-Light sometimes outperforms the TurbNetGeo-Light, but it is not clear that it is necessarily better. The provided examples are only a small set of samples that are available for comparison and were chosen to highlight problems that may occur for the network prediction. Even for the 'bad' category, the predictions are not completely unusable, but there are edge cases where the prediction would be dangerous to use in place of the high-fidelity solver. However, the CNN is suitable as an initial pre-processing step for the high-fidelity optimisation from Chapter 7, where the only consequence is perhaps removing a viable GWHP from the hypothetical GWHP list, or performing extra outer iterations. Therefore, the recommended network is the TurbNetGeo-Light with 32 initial features, followed by TurbNetGeo-NoSkip-Light with 32 initial features.

8.7 Recommendations for Future Work

The objective of the surrogate model development was to evaluate whether deep learning is a suitable candidate to develop a low-fidelity subsurface GWHP temperature plume prediction tool and to show the viability of the proposed deep learning method. The initial deep learning network is a simple construction which proved that using already available Darcy flow velocities is a viable candidate for training a neural network. The Darcy velocity can be generated on a small domain as in this study or scaled up to large 3D domains to obtain more accurate simulation training and testing data. However, shortcomings of the initial design and development performed in this thesis can be improved by

1. rotating all input and output data samples, such that the thermal plume flows in the same direction, i.e., rotate the domain such that the plume flows from left to right at the centre point and has no $\boldsymbol{q}_y$ velocity component at the centre. Therefore, the network does not have to learn the initial direction that the plume must propagate towards, but only how to morph with the Darcy flow streamlines and the temperature magnitude,
2. generating more training data which can result in more accurate models, but with longer training times,
3. adding the PDE constraints to the loss function in addition to a limited dataset size, especially if larger numerical simulations are too expensive to generate sufficient training data,
4. performing further hyper-parameter optimisation such as varying the number of layers, learning rate, batch size and number of training epochs,
5. extending the domain to 3D to account for vertical flow effects,
6. training separate networks for different scenarios, such as high Darcy and low Darcy velocity at the GWHP, or near-field and far-field networks.

In addition to the improvements for a single GWHP prediction, training a network to predict the influence of multiple interacting GWHPs would be able to create a more accurate prediction tool. This would require a network to predict the new thermal plume from a non-uniform background temperature profile. This would greatly enhance the usability and applicability of the surrogate model.

8.8 Summary of Chapter 8

Surrogate models are required to determine the thermal influence of thousands of interacting GWHPs in a fast and efficient manner. Within the last decade, deep learning has emerged as a ground-breaking method to learn complex functions using only input and output data. However, deep learning had not been applied to predicting the thermal plume from shallow GWHPs.

In this thesis, we developed a novel method to use only the subsurface Darcy velocity to predict the temperature plume of a shallow GWHP. Convolutional neural networks were constructed that accepted the Darcy velocity as input channels and calculated the temperature field as an output. A total of 800 datasets were generated to train and test the network.

The networks demonstrated the ability of deep learning to be a viable solution for building fast and accurate surrogate models. The network was able to predict the general direction of the thermal plume for almost all testing samples, as well as the morphing of the plume for many samples. However, the network is unable to provide an accurate prediction for all samples and more development of the network is required to enhance the accuracy.

9 Conclusion

This thesis has contributed to the improvement of state-of-the-art methods in partitioned coupling for multi-physics and multi-component simulations. These contributions were developed as part of two projects: the preDOM project and the GEO.KW project. The research for both projects resulted in splitting this thesis in two parts:

- Part I: analysis and development of numerical methods for partitioned multi-physics simulation coupling.
- Part II: developing a novel coupling method for large scale, shallow geothermal energy infrastructure optimisation.

Although each of these themes are different, they both focus on overcoming the challenges of performing more complex partitioned simulation coupling.

9.1 Summary of the Thesis

The individual contributions from both Part I and Part II are discussed below, with the central focus on extending the field of knowledge for partitioned simulation coupling. A summary of each part is provided below.

9.1.1 Part I: Numerical Methods for Partitioned Simulation Coupling

The aim of Part I was to analyse the current state-of-the-art numerical methods for partitioned simulation coupling with preCICE, a general purpose partitioned library used to perform surface coupled multi-physics simulations. The specific focus was to improve the usability of preCICE, while reducing the computational cost and improving robustness of the underlying numerical algorithms. The two areas in preCICE that were identified for improvements were the quasi-

Newton coupling acceleration and the radial basis function data mapping method. Both areas can significantly impact the simulation runtime of the coupling update step, especially in cases where the user-specified input parameters lead to an ill-configured numerical coupling setup. These "bad" input parameters may lead to longer simulation times or even instability in the simulation coupling. Defining a set of suitable input parameters is non-trivial and is often problem dependent, requiring experience from the user.

The largest expense in the quasi-Newton update step is a QR-decomposition. This is performed in each coupling iteration as a result of repeated re-scaling of different physical quantities in the coupling vector. In Chapter 3, the quasi-Newton method was enhanced by developing a combination of pre-scaling weight monitoring method and a new QR3 filter, which, when combined can significantly reduce the number of complete QR-decompositions performed. The improved methods were tested for four different multi-physics test cases, where a reduction in the runtime of the update step was seen for all cases. In extreme cases, the cost of the QR-decomposition step was reduced by up to 2 orders of magnitude. As the computational cost of the update step was no longer a limit on the input configurations, various input parameters were tested for each test case, culminating in a list of suggested default values to be used by inexperienced users or for new multi-physics simulations with unknown behaviour.

The radial basis function mapping method in preCICE involves a large linear system, that can either be solved using direct or iterative methods. The direct method limits the number of vertices on the coupling interface due to its computational expense. Iterative methods circumvent expensive direct decompositions of the system matrix, but results in potentially expensive inner mapping iterations in each coupling iteration and an increased error due to the additional solver tolerance. In Chapter 4, a partition-of-unity method was developed in PyRBF which decomposes the mesh into many smaller, independent interpolation sub-domains. The interpolation solution on the global output mesh is formed by a summation of all solutions from all sub-domains and applying a weighting function to vertices that cover multiple sub-domains. The interpolation accuracy and scalability were evaluated for a 2D surface and a 3D box volume domain. The partition-of-unity method was able to significantly reduce the compute mapping time while providing accurate data mapping for surface and volume coupled problems. In addition, the value of the radial basis functions' support radius was found to influence the interpolation accuracy more than the number of sub-domains or the number of vertices within each sub-domain. Therefore, increasing the number of sub-domains and using large support radii or even a global basis function is better than having to restrict the support radius for large input meshes and provides a simple default configuration for coupled multi-physics data mapping.

9.1.2 Part II: Geothermal Energy Infrastructure Optimisation

The focus in Part II of this thesis was to develop a partitioned coupling method to enable the optimisation of shallow groundwater heat pumps (GWHP) throughout the city of Munich. Heating and cooling of buildings using GWHPs may help reduce our dependence on fossil fuel based sources as well as alleviate the subsurface heat island effect. A novel partitioned simulation-optimisation coupling scheme was developed in Chapter 6, that combines a numerical groundwater solver with an energy infrastructure optimisation solver. The coupled approach allows for the optimisation solver to account for the influence that each GWHP has on the subsurface temperature and on the pressure conditions. The numerical groundwater simulation was performed with PFLOTRAN and the energy infrastructure optimisation was performed with urbs.

Due to the size of the groundwater model of the entire city of Munich, the city was decomposed into smaller PFLOTRAN models. The urbs energy infrastructure model was decomposed into multiple urbs sub-domains within each PFLOTRAN region. The urbs model decomposition allowed for the implementation of a novel inner-outer iteration coupling approach, a so-called staggered coupling. This allowed for each urbs region to add one new GWHP to the cost-optimal solution in every outer iteration, while multiple inner iterations ensured that a converged groundwater solution was achieved before checking if the GWHPs were operating within a pre-determined range. The staggered coupling method was applied to two test cases in Chapter 7: the COM-4 model and the REG-30 model. In both test cases, the staggered coupling approach allowed for GWHPs to be added to the cost-optimal solution in successive outer iterations, all while remaining within the defined operating range. The REG-30 model was modified to influence the selection of GWHPs, which showed that the staggered coupling approach provides a highly complex platform for GWHP optimisation.

In Chapter 8, a novel deep learning based surrogate model was developed to predict the temperature field around a GWHP injection well. A convolutional neural network was trained to predict the thermal plume that propagates downstream from an injection well, using only the Darcy velocity field in the local area surrounding the injection well. The developed method can be used to create a surrogate for any location in the city, as the Darcy velocity field is available for the entire city of Munich. The online evaluation phase only requires extracting the local Darcy velocities, without having to perform any expensive numerical simulation to generate input data.

9.2 Contributions

The various contributions to the field of partitioned simulation coupling are summarised below.

Contribution 1. *Extended capabilities of the quasi-Newton method for multi-physics simulation coupling.*

Quasi-Newton methods improve the convergence rate of partitioned multi-physics simulation coupling when combined with additional numerical methods such as pre-scaling and filtering. The new algorithms presented reduced the computational cost by orders of magnitude, while still maintaining a fast rate of convergence.

Contribution 2. *Extended capabilities of RBF interpolation for surface coupling data mapping for multi-physics simulations.*

Reducing the radial basis function interpolation error is typically performed by optimising the basis function support radius. Extensive interpolation tests, using the partition-of-unity method, showed that the interpolation error is less sensitive to the number of sub-domains than to the support radius size. The partition-of-unity method allowed for large support radii or even global support of basis functions to be used with direct solvers to provide a fast, scalable and accurate interpolation solver.

Contribution 3. *Novel partitioned coupling method for shallow geothermal energy infrastructure optimisation.*

A novel staggered coupling procedure was developed, enabling communication between an energy infrastructure optimisation software and a numerical subsurface flow software. The coupling scheme and data exchange method developed in this work allowed for cost-optimal solution of the GWHP usage to be determined while remaining within operational constraints. The coupling method was tested on multiple models, providing new insights into the optimal GWHP usage on a city-wide scale.

Contribution 4. *Novel deep learning surrogate model to predict groundwater temperatures from easily attainable input data.*

A 2D convolutional neural network was trained to predict the GWHP thermal plume using only the Darcy velocity vectors as input. As the thermal plume follows the direction of the subsurface fluid, this provided enough information for the neural network to predict the plume with sufficient accuracy. This method allows for more complex networks to be built using this principle as the 3D Darcy velocity data already exists for the Munich subsurface.

9.3 Limitations

Despite the contributions made in this work, a few limitations persist.

The quasi-Newton improvements were tested on only a small number of computing ranks for small but numerically difficult test cases. The runtime savings of the improved algorithms may be less pronounced for highly scalable solvers running in parallel.

The partition-of-unity radial basis function interpolation was evaluated on a simple surface and volume domain with varying mesh sizes. The sub-domain decomposition is performed on the main rank and requires that each sub-domain is of equal size. This uniform decomposition may end up with empty ranks that do not have any vertices and may lead to poor load-balancing in real-world cases.

The staggered coupling optimisation was performed for one PFLOTRAN region only and limited to two regions per rank for the urbs optimisation. Extending the staggering procedure to allow for any number of regions to be run on a single rank, with an arbitrary number of downstream regions may allow for more complicated urbs region layouts.

The convolutional neural network of Chapter 8 is limited to 2D square domains. The network was only trained for GWHPs operating at a mass flow rate of $0.05\ell/s$ and limited to predicting the thermal plumes for GWHPs operating at this setting. The network does not account for the interaction between multiple thermal plumes.

9.4 Future Work

Based on the results achieved in this thesis, the following research topics are suggested:

- The capabilities of the pre-scaling weight monitoring method, combined with the QR3 filter, should be tested on multi-physics problems with more than 2 solvers. Extending the number of sub-vectors may require more complete QR-decompositions due to a change in the pre-scaling weights. Larger test cases may find the limit of the computational runtime improvements.
- The partition-of-unity method should be implemented in preCICE. As the mesh is already decomposed by the individual solvers, a sub-domain partitioning would need to be implemented on each rank. This would allow full control of the number of vertices per sub-domain and subsequently, control over the runtime of the interpolation and the evaluation step.
- The staggered coupling approach should be applied to multiple PFLOTRAN regions throughout the city to find a complete, cost-optimal solution for GWHP usage in the city of Munich. The procedure remains the same as presented in Part II but requires more computational time. A multi-fidelity method, that uses a surrogate model to remove hypothetical GWHPs that are immediately upstream of an existing GWHP, should be implemented to reduce the potential selection of obviously bad GWHPs.
- Improved deep learning models should be developed to improve the surrogate model solution, including 3D effects. A larger data set should be generated from simulation results to train the network and combined with a physics-informed neural network (PINN) loss to limit the number of numerical simulations required to generate input data. PINNs can extend the range of available training data to use currently accessible Darcy velocity information in the entire city of Munich, without needing the actual temperature plume information. Furthermore, the network should be trained to handle non-uniform background temperatures to be able to model multiple interacting plumes.

Bibliography

[Agg18] **Aggarwal**, C. C.: *Neural Networks and Deep Learning*, 1st ed., Springer Cham, 2018, doi:https://doi.org/10.1007/978-3-319-94463-0

[Ali18] **Alikhani**, J. et al.: *Finding a good shape parameter of RBF to solve PDEs based on the particle swarm optimization algorithm*, Alexandria Engineering Journal 57.4, 2018, pp. 3641–3652, ISSN: 1110-0168, doi:10.1016/j.aej.2017.11.024, https://doi.org/10.1016/j.aej.2017.11.024

[All03] **Allen**, A.; **Milenic**, D.; **Sikora**, P.: *Shallow gravel aquifers and the urban heat island effect: a source of low enthalpy geothermal energy*, Geothermics 32.4, 2003, Selected Papers from the European Geothermal Conference 2003, pp. 569–578, ISSN: 0375-6505, doi:https://doi.org/10.1016/S0375-6505(03)00063-4, https://www.sciencedirect.com/science/article/pii/S0375650503000634

[And65] **Anderson**, D.: *Iterative procedures for nonlinear integral equations*, Journal of the ACM 12.4, 1965, pp. 547–560

[Ans14] **Ansari**, E.: *Development of a surrogate simulator for two-phase subsurface flow simulation using trajectory piecewise linearization.* J Petrol Explor Prod Technol, 2014, doi:https://doi.org/10.1007/s13202-013-0084-8

[Ash15] **Asher**, M. et al.: *A review of surrogate models and their application to groundwater modeling*, Water Resour. Res. 51, 2015, pp. 5957–5973, doi:10.1002/2015WR016967

[Att20] **Attard**, G. et al.: *A novel concept for managing thermal interference between geothermal systems in cities*, Renewable Energy 145, 2020, pp. 914–924, doi:10.1016/j.renene.2019.06.095, https://doi.org/10.1016/j.renene.2019.06.095

[Aya15] **Ayachit**, U.: *The ParaView Guide: A Parallel Visualization Application*, Kitware, 2015

[Bal22] **Balay**, S. et al.: *PETSc Web page*, https://petsc.org/, 2022, https://petsc.org/

[Ban09] **Banks**, D.: *An introduction to thermogeology and the exploitation of ground source heat*, Quarterly Journal of Engineering Geology and Hydrogeology, 2009, doi:https://doi.org/10.1144/1470-9236/08-077

[Bay12] **Bayer**, P. et al.: *Greenhouse gas emission savings of ground source heat pump systems in europe: a review*, Renewable and Sustainable Energy Reviews, 2012

[Bec13] **Beck**, M. et al.: *Geometric arrangement and operation mode adjustment in low-enthalpy geothermal borehole fields for heating*, Energy 49.1, 2013, pp. 434–443, doi:10.1016/j.energy.2012.10.060, http://dx.doi.org/10.1016/j.energy.2012.10.060

[Bia16] **Biazar**, J.; **Hosami**, M.: *Selection of an interval for variable shape parameter in approximation by radial basis functions*, 1, 2016

[Bia17] **Biazar**, J.; **Hosami**, M.: *An interval for the shape parameter in radial basis function approximation*, Applied Mathematics and Computation 315, 2017, pp. 131–149, ISSN: 00963003, doi:10.1016/j.amc.2017.07.047, http://dx.doi.org/10.1016/j.amc.2017.07.047

[Bog14] **Bogaers**, A. et al.: *Quasi-Newton methods for implicit black-box FSI coupling*, Comput. Methods Appl. Mech. Engrg. 279, 2014, pp. 113–132, ISSN: 0045-7825, doi:10.1016/j.cma.2014.06.033, http://dx.doi.org/10.1016/j.cma.2014.06.033

[Bog16a] **Bogaers**, A. et al.: *An evaluation of quasi-newton methods for application to fsi problems involving free surface flow and solid body contact*, Comput. Struct. 173, 2016, pp. 71–83, doi:https://doi.org/10.1016/j.compstruc.2016.05.018

[Bog16b] **Bogaers**, A.: *Efficient and robust partitioned solution schemes for fluid-structure interactions*, Journal of Numerical Analysis and Approximation Theory 45.1, 2016, pp. 84–86

[Bon13] **Bonte**, M.; **van Breukelen**, B. M.; **Stuyfzand**, P. J.: *Temperature-induced impacts on groundwater quality and arsenic mobility in anoxic aquifer sediments used for both drinking water and shallow geothermal energy production*, Water Research 47.14, 2013, pp. 5088–5100, ISSN: 0043-1354, doi:https://doi.org/10.1016/j.watres.2013.05.049, https://www.sciencedirect.com/science/article/pii/S004313541300479X

[Böt22] **Böttcher**, F.; **Zosseder**, K.: *Thermal influences on groundwater in urban environments a multivariate statistical analysis of the subsurface heat island effect in munich*, Science of The Total Environment 810, 2022, p. 152193, ISSN: 0048-9697, doi:https://doi.org/10.1016/j.scitotenv.2021.152193, https://www.sciencedirect.com/science/article/pii/S0048969721072697

[Boy14] **Boyce**, S. E.; **Yeh**, W. W.: *Parameter-independent model reduction of transient groundwater flow models: Application to inverse problems*, Advances in Water Resources 69, 2014, pp. 168–180, ISSN: 03091708, doi:10.1016/j.advwatres.2014.04.009, http://dx.doi.org/10.1016/j.advwatres.2014.04.009

[Bra06] **Brand**, M.: *Fast low-rank modifications of the thin singular value decomposition*, Linear Algebra and its Applications 415.1, 2006, Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems, pp. 20–30, ISSN: 0024-3795, doi:https://doi.org/10.1016/j.laa.2005.07.021, https://www.sciencedirect.com/science/article/pii/S0024379505003812

[Bre07] **Brenk**, M.: *Algorithmische Aspekte der Fluid-Struktur-Wechselwirkung auf kartesischen Gittern.* PhD thesis, Universität Stuttgart, 2007

[Buh03] **Buhr**, W.: *What is infrastructure*, tech. rep., Department of Economics, School of Economic Disciplines, University of Siegen, 2003, https://www.wiwi.uni-siegen.de/vwl/research/diskussionsbeitraege/pdf/107-03.pdf

[Car09] **Cardoso**, M. A.; **Durlofsky**, L. J.; **Sarma**, P.: *Development and application of reduced-order modeling procedures for subsurface flow simulation*, International Journal for Numerical Methods in Engineering 77.9, 2009, pp. 1322–1350, doi:https://doi.org/10.1002/nme.2453, eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2453, https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2453

[Cav07] **Cavagna**, L.; **Quaranta**, G.; **Mantegazza**, P.: *Application of NavierStokes simulations for aeroelastic stability assessment in transonic regime*, Computers Structures 85.11, 2007, pp. 818–832, doi:https://doi.org/10.1016/j.compstruc.2007.01.005

[Cav16] **Cavoretto**, R.; **Rossi**, A. D.; **Perracchione**, E.: *Efficient computation of partition of unity interpolants through a block-based searching technique*, Computers and Mathematics with Applications 71.12, 2016, pp. 2568–2584, ISSN: 0898-1221, doi:10.1016/j.camwa.2016.04.021, http://dx.doi.org/10.1016/j.camwa.2016.04.021

[Cav18] **Cavoretto**, R.; **Rossi**, A. D.; **Perracchione**, E.: *Optimal Selection of Local Approximants in RBF-PU Interpolation*, January, 2018, doi:10.1007/s10915-017-0418-7, arXiv: arXiv:1703.04282v1

[Cav20] **Cavoretto**, R.; **Rossi**, A. D.: *An adaptive LOOCV-based refinement scheme for RBF collocation methods over irregular domains An adaptive LOOCV-based refinement scheme for RBF collocation methods over irregular domains*, January, 2020, doi:10.1016/j.aml.2019.106178

[Che12] **Cheng**, A. H.: *Multiquadric and its shape parameterA numerical investigation of error estimate, condition number, and round-off error by arbitrary precision computation*, Engineering Analysis With Boundary Elements 36.2, 2012, pp. 220–239, ISSN: 0955-7997, doi:10.1016/j.enganabound.2011.07.008, http://dx.doi.org/10.1016/j.enganabound.2011.07.008

[Cho22] **Chourdakis**, G. et al.: *Precice v2: a sustainable and user-friendly coupling library [version 1; peer review: 2 approved]*, Open Res Europe 2:51, 2022, doi:https://doi.org/10.12688/openreseurope.14445.1)

[Con19] **Conrad**, J.; **Greif**, S.: *Modelling load profiles of heat pumps*, Energies 12.4, 2019, ISSN: 1996-1073, doi:10.3390/en12040766, https://www.mdpi.com/1996-1073/12/4/766

[Cou18] **Council of the European Union**: *Directive (eu) 2018/2001 of the european parliament and of the council of 11 december 2018 on the promotion of the use of energy from renewable sources*, Official Journal of the European Union, 2018, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32018L2001

[Dan76] **Daniel**, J. et al.: *Reorthogonalization and stable algorithms for updating the gram-schmidt factorization*, Mathematics of Computation 30.136, 1976, pp. 772–795

[Dav22a] **Davis**, K.; **Schulte**, M.: *COM-4 model to replicate simulation results for the GEO.KW project*, version V1, 2022, doi:10.18419/darus-3185, https://doi.org/10.18419/darus-3185

[Dav22b] **Davis**, K.; **Schulte**, M.: *REG-30 model to replicate simulation results for the GEO.KW project*, version V1, 2022, doi:10.18419/darus-3195, https://doi.org/10.18419/darus-3195

[Dav22c] **Davis**, K.; **Schulte**, M.: *Replication Data for: Geothermal-ML - predicting thermal plume from groundwater heat pumps*, version V1, 2022, doi:10.18419/darus-3184, https://doi.org/10.18419/darus-3184

[Dav22d] **Davis**, K.; **Schulte**, M.: *Replication Data for: Radial basis function interpolation with partition of unity for PyRBF*, version V1, 2022, doi:10.18419/darus-3183, https://doi.org/10.18419/darus-3183

[De 12] **De Paly**, M. et al.: *Optimization of energy extraction for closed shallow geothermal systems using linear programming*, Geothermics 43, 2012, pp. 57–65, doi:10.1016/j.geothermics.2012.03.001, http://dx.doi.org/10.1016/j.geothermics.2012.03.001

[Deg09] **Degroote**, J.; **Bathe**, K.-j.; **Vierendeels**, J.: *Performance of a new partitioned procedure versus a monolithic procedure in fluid structure interaction*, Computers and Structures 87.11-12, 2009, pp. 793–801, ISSN: 0045-7949, doi:10.1016/j.compstruc.2008.11.013, http://dx.doi.org/10.1016/j.compstruc.2008.11.013

[Dho04] **Dhondt**, G.: *The Finite Element Method for Three-Dimensional Thermomechanical Applications*, John Wiley & Sons, Ltd, 2004, isbn:9780470857526, doi:10.1002/0470021217

[Doh15] **Doherty**, J.: *Calibration and Uncertainty Analysis for Complex Environmental Models*, Brisbane, Australia: Watermark Numerical Computing, 2015, isbn:978-0-9943786-0-6

[Dor15] **Dorfner2015**: *Open Source Modelling and Optimisation of Energy Infrastructure at Urban Scale*, PhD thesis, Technical University of Munich, 2015

[Dor20] **Dorfner**, J.: *urbs: A linear optimisation model for distributed energy systems*, 2020, https://urbs.readthedocs.io/en/latest/index.html

[du 18] **du Toit**, P.: *An artificial intelligence approach for biomass devolatilisation in an industrial CFD model with advanced turbulence-chemistry interaction*, PhD thesis, Stellenbosch University, 2018

[Duc15] **Duchaine**, F. et al.: *Analysis of high performance conjugate heat transfer with the OpenPALM coupler*, Computational Science &amp Discovery 8.1, 2015, p. 015003, doi:10.1088/1749-4699/8/1/015003, https://doi.org/10.1088/1749-4699/8/1/015003

[Dur19] **Duraisamy**, K.; **Iaccarino**, G.; **Xiao**, H.: *Turbulence modeling in the age of data*, Annual Review of Fluid Mechanics 51.1, 2019, pp. 357–377, doi:10.1146/annurev-fluid-010518-040547, eprint: https://doi.org/10.1146/annurev-fluid-010518-040547, https://doi.org/10.1146/annurev-fluid-010518-040547

[Eur11] **European Environment Agency**: *Communication from the commission to the european parliament, the council, the european economic and social committee and the committee of the regions: energy efficiency plan 2011*, tech. rep., European Environment Agency, 2011, https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2011:0109:FIN:EN:PDF

[Fan09] **Fang**, H. R.; **Saad**, Y.: *Two classes of multisecant methods for nonlinear acceleration*, Numerical Linear Algebra with Applications 16.3, 2009, pp. 197–221, ISSN: 10705325, doi:10.1002/nla.617

[Fas07] **Fasshauer**, G. E.; **Zhang**, J. G.: *On choosing optimal shape parameters for RBF approximation*, 2007, pp. 345–368, doi:10.1007/s11075-007-9072-8

[Fer07] **Ferguson**, G.; **Woodbury**, A. D.: *Urban heat island in the subsurface*, Geophysical Research Letters 34.23, 2007, doi:https://doi.org/10.1029/2007GL032324, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2007GL032324

[För07] **Förster**, C.; **Wall**, W.; **Ramm**, E.: *Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows.* Comput. Methods Appl. Mech. Eng. 196, 2007, pp. 1278–1293, doi:https://doi.org/10.1016/j.cma.2006.09.002

[Gao21] **Gao**, H.; **Sun**, L.; **Wang**, J. X.: *Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels*, Physics of Fluids 33.7, 2021, ISSN: 10897666, doi:10.1063/5.0054312

[Gar18a] **García-Gil**, A. et al.: *Decreased waterborne pathogenic bacteria in an urban aquifer related to intense shallow geothermal exploitation*, Science of The Total Environment 633, 2018, pp. 765–775, ISSN: 0048-9697, doi:https://doi.org/10.1016/j.scitotenv.2018.03.245, https://www.sciencedirect.com/science/article/pii/S0048969718310040

[Gar18b] **Garmanjani**, G.; **Cavoretto**, R.; **Esmaeilbeigi**, M.: *A rbf partition of unity collocation method based on finite difference for initialboundary value problems*, Computers Mathematics with Applications 75.11, 2018, pp. 4066–4090, ISSN: 0898-1221, doi:https://doi.org/10.1016/j.camwa.2018.03.014, https://www.sciencedirect.com/science/article/pii/S0898122118301433

[Gar20] **García-Gil**, A. et al.: *Nested shallow geothermal systems*, Sustainability (Switzerland) 12.12, 2020, doi:10.3390/su12125152

[Gat15] **Gatzhammer**, B.: *Efficient and Flexible Partitioned Simulation of Fluid-Structure Interactions.* PhD thesis, Technische Universität München, 2015

[Ger03] **Gerbeau**, J.; **Vidrascu**, M.: *A quasi-newton algorithm based on a reduced model for fluid-structure interaction problems in blood flows*, tech. rep., 2003, https://hal.inria.fr/inria-00071895

[Geu09] **Geuzaine**, C.; **Remacle**, J.: *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*, International Journal for Numerical Methods in Engineering, 2009

[Gla] **Glaessgen**, E.; **Stargel**, D.: *The digital twin paradigm for future nasa and us air force vehicles*, In 53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference, https://ntrs.nasa.gov/citations/20120008178

[Gri17] **Grieves**, M.; **Vickers**, J.: *Digital twin: mitigating unpredictable, undesirable emergent behavior in complex systems*, Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches, ed. by **Kahlen**, F.-J.; **Flumerfelt**, S.; **Alves**, A., Springer International Publishing, 2017, pp. 85–113, isbn:978-3-319-38756-7, doi:10.1007/978-3-319-38756-7_4, https://doi.org/10.1007/978-3-319-38756-7_4

[Gue10] **Guennebaud**, G.; **Jacob**, B., et al.: *Eigen v3*, http://eigen.tuxfamily.org, 2010

[Gun11] **Gunawardhana**, L.; **Kazama**, S.; **Kawagoe**, S.: *Impact of urbanization and climate change on aquifer thermal regimes*, Water Resour Manage, 2011, doi:https://doi.org/10.1007/s11269-011-9854-6

[Gup22] **Gupta**, R.; **Jaiman**, R.: *A hybrid partitioned deep learning methodology for moving interface and fluidstructure interaction*, Computers Fluids 233, 2022, p. 105239, ISSN: 0045-7930, doi:https://doi.org/10.1016/j.compfluid.2021.105239, https://www.sciencedirect.com/science/article/pii/S0045793021003479

[Hae16] **Haelterman**, R. et al.: *Improving the performance of the partitioned QN-ILS procedure for fluid-structure interaction problems: Filtering*, Computers and Structures 171, 2016, pp. 9–17, ISSN: 00457949, doi:10.1016/j.compstruc.2016.04.001, http://dx.doi.org/10.1016/j.compstruc.2016.04.001

[Hal22] **Halilovic**, S.; **Odersky**, L.; **Hamacher**, T.: *Integration of groundwater heat pumps into energy system optimization models*, Energy 238, 2022, p. 121607, ISSN: 0360-5442, doi: https://doi.org/10.1016/j.energy.2021.121607, https://www.sciencedirect.com/science/article/pii/S0360544221018557

[Ham14] **Hammond**, G.; **Lichtner**, P.; **Mills**, R.: *Evaluating the performance of parallel subsurface simulators: an illustrative example with pflotran*, Water Resources Research 50, 2014, pp. 208–228, doi:10.1002/2012WR013483

[Hem19] **Hemmerle**, H. et al.: *Estimation of groundwater temperatures in paris, france*, Geofluids, 2019, doi:https://doi.org/10.1155/2019/5246307

[Hij20] **Hijazi**, S. et al.: *Data-driven pod-galerkin reduced order model for turbulent flows*, Journal of Computational Physics 416, 2020, p. 109513, ISSN: 0021-9991, doi:https://doi.org/10.1016/j.jcp.2020.109513, https://www.sciencedirect.com/science/article/pii/S0021999120302874

[Hsu12] **Hsu**, M.-C.: *Fluid-Structure Interaction Analysis of Wind Turbines*, PhD thesis, University of California, San Diego, 2012, p. 196

[Iro69] **Irons**, B.; **Tuck**, R.: *A version of the aitken accelerator for compute iterations*, International Journal for Numerical Methods in Engineering 1, 1969, pp. 275–277

[Jat16] **Jatnieks**, J. et al.: *Data-driven surrogate model approach for improving the performance of reactive transport simulations*, Energy Procedia 97, 2016, pp. 447–453, ISSN: 1876-6102, doi:10.1016/j.egypro.2016.10.047, http://dx.doi.org/10.1016/j.egypro.2016.10.047

[Joh22] **Johnson**, E. et al.: *Effects of membrane and flexural stiffnesses on aortic valve dynamics: identifying the mechanics of leaflet flutter in thinner biological tissues*, Forces in Mechanics 6, 2022, p. 100053, ISSN: 2666-3597, doi:https://doi.org/10.1016/j.finmec.2021.100053, https://www.sciencedirect.com/science/article/pii/S2666359721000445

[Kam16] **Kamaludeen**, S. M. K.; **Zuijle**, A. van; **Bijl**, H.: *Surrogate based wind farm layout optimization using manifold mapping*, Journal of Physics: Conference Series 753, 2016, p. 092005, doi: 10.1088/1742-6596/753/9/092005, https://doi.org/10.1088/1742-6596/753/9/092005

[Kan19] **Kani**, J. N.; **Elsheikh**, A. H.: *Reduced-Order Modeling of Subsurface Multi-phase Flow Models Using Deep Residual Recurrent Neural Networks*, Transport in Porous Media 126.3, 2019, pp. 713–741, ISSN: 1573-1634, doi:10.1007/s11242-018-1170-7, https://doi.org/10.1007/s11242-018-1170-7

[Kar19] **Karageorghis**, A.: *Variable shape parameter Kansa RBF method for the solution of nonlinear boundary value problems*, February, 2019, doi:10.1016/j.enganabound.2019.02.005

[Kas21] **Kashefi**, A.; **Rempe**, D.; **Guibas**, L. J.: *A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries*, Physics of Fluids 33.2, 2021, ISSN: 10897666, doi:10.1063/5.0033376

[Key12] **Keyes**, D. et al.: *Multiphysics Simulations: Challenges and Opportunities.* Tech. rep., Argonne National Laboratory, 2012

[Kin14] **Kingma**, D. P.; **Ba**, J.: *Adam: a method for stochastic optimization*, 2014, doi:10.48550/ARXIV.1412.6980, https://arxiv.org/abs/1412.6980

[Kin87] **Kinzelbach**, W.: *Numerische Methoden zur Modellierung des Transports von Schadstoffen im Grundwasser*, 2nd ed., Munich: Oldenbourg, 1987

[Kor14] **Korobenko**, A.: *Advanced FluidStructure Interaction Techniques in Application to Horizontal and Vertical Axis Wind Turbines*, PhD thesis, University of California, San Diego, 2014, p. 93

[Küt08] **Küttler**, U.; **Wall**, W.: *Fixed-point fluidstructure interaction solvers with dynamic relaxation*, Comput Mech 43, 2008, pp. 61–72, doi:https://doi.org/10.1007/s00466-008-0255-5

[Lau17] **Laubscher**, R.: *Utilization of artificial neural networks to resolve chemical kinetics in turbulent fine structures of an advanced CFD combustion model*, PhD thesis, Stellenbosch University, 2017

[Lau21] **Laubscher**, R.: *Simulation of multi-species flow and heat transfer using physics-informed neural networks Simulation of multi-species flow and heat transfer using physics-informed neural networks*, Phys. Fluids 087101.July, 2021, doi:10.1063/5.0058529

[Leu09] **Leuschner**, U.: *Die deutsche gasversorgung von den anfängen bis 1998*, 2009, http://www.udo-leuschner.de/pdf/gasversorgung.pdf.

[Li13] **Li**, X. et al.: *Model reduction of a coupled numerical model using proper orthogonal decomposition*, Journal of Hydrology 507, 2013, pp. 227–240, ISSN: 0022-1694, doi:https://doi.org/10.1016/j.jhydrol.2013.09.011, https://www.sciencedirect.com/science/article/pii/S0022169413006562

[Li16] **Li**, S. et al.: *A Meshless Radial Basis Function Based on Partition of Unity Method for Piezoelectric Structures*, Mathematical Problems in Engineering 2016, 2016

[Lin15] **Lindner**, F. et al.: *A comparison of various quasi-newton schemes for partitioned fluid-structure interaction*, In Procedings of the VI International Conference on Computational Methods for Coupled Problems in Science and Engineering, Venice, Italy, 2015, pp. 477–488

[Lin16] **Ling**, J.; **Kurzawski**, A.; **Templeton**, J. A.: *Reynolds averaged turbulence modelling using deep neural networks with embedded invariance*, Journal of Fluid Mechanics 807, 2016, pp. 155–166

[Lin19] **Lindner**, F.: *Data Transfer in Partitioned Multi-Physics Simulations: Interpolation & Communication*, PhD thesis, Universität Stuttgart, 2019

[Mar08] **Marks**, L. D.; **Luke**, D. R.: *Robust mixing for ab initio quantum mechanical calculations*, Physical Review B - Condensed Matter and Materials Physics 78.7, 2008, pp. 1–12, ISSN: 10980121, doi:10.1103/PhysRevB.78.075114, arXiv: 0801.3098

[Men13a] **Menberg**, K. et al.: *Subsurface urban heat islands in german cities*, Science of the Total Environment, 2013

[Men13b] **Menberg**, K. et al.: *Long-term evolution of anthropogenic heat fluxes into a subsurface urban heat island*, Environmental Science & Technology 47.17, 2013, pp. 9747–9755, doi:10.1021/es401546u, eprint: https://doi.org/10.1021/es401546u, https://doi.org/10.1021/es401546u

[Men19] **Meng**, B. et al.: *Evaluating the thermal impacts and sustainability of intensive shallow geothermal utilization on a neighborhood scale : Lessons learned from a case study*, Energy Conversion and Management 199.July, 2019, p. 111913, doi:10.1016/j.enconman.2019.111913, https://doi.org/10.1016/j.enconman.2019.111913

[Mil05] **Miller**, K.: *Nonlinear krylov and moving nodes in the method of lines*, Journal of Computational and Applied Mathematics 183.2, 2005, pp. 275–287, doi:https://doi.org/10.1016/j.cam.2004.12.032, https://www.sciencedirect.com/science/article/pii/S0377042705000658

[Mil20] **Milovanovi**, S.; **von Sydow**, L.: *A high order method for pricing of financial derivatives using radial basis function generated finite differences*, Mathematics and Computers in Simulation 174, 2020, pp. 205–217, ISSN: 0378-4754, doi:https://doi.org/10.1016/j.matcom.2020.02.005, https://www.sciencedirect.com/science/article/pii/S0378475420300446

[Muk19] **Mukhametzhanov**, R. C. A. D. R. M. S.; **Sergeyev**, Y. D.: *On the search of the shape parameter in radial basis functions using univariate global optimization methods*, Journal of Global Optimization, 2019, ISSN: 1573-2916, doi:10.1007/s10898-019-00853-3, https://doi.org/10.1007/s10898-019-00853-3

[Mül14] **Müller**, N.; **Kuttler**, W.; **Barlag**, A.: *Analysis of the subsurface urban heat island in oberhausen, germany*, Clim Res, 2014, doi:https://doi.org/10.3354/cr01195

[Nan21] **Nannini**, G. et al.: *Aortic hemodynamics assessment prior and after valve sparing reconstruction: a patient-specific 4d flow-based fsi model*, Computers in Biology and Medicine 135, 2021, p. 104581, ISSN: 0010-4825, doi:https://doi.org/10.1016/j.compbiomed.2021.104581, https://www.sciencedirect.com/science/article/pii/S0010482521003759

[Oos00] **Oosterlee**, C. W.; **Washio**, T.: *Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows*, SIAM Journal on Scientific Computing 21.5, 2000, pp. 1670–1690, doi:10.1137/S1064827598338093, https://doi.org/10.1137/S1064827598338093

[Pan14] **Pan**, I. et al.: *Artificial Neural Network based surrogate modelling for multi- objective optimisation of geological CO 2 storage operations*, Energy Procedia 63, 2014, pp. 3483–3491, ISSN: 1876-6102, doi:10.1016/j.egypro.2014.11.377, http://dx.doi.org/10.1016/j.egypro.2014.11.377

[Par13] **Parolini**, N.; **Lombardi**, M.: *Unsteady FSI simulation of downwind sails.* MARINE V. "MARINE V : proceedings of the V International Conference on Computational Methods in Marine Engineering", 2013, pp. 266–277

[Pas11] **Pasetto**, D.; **Guadagnini**, A.; **Putti**, M.: *Pod-based monte carlo approach for the solution of regional scale groundwater flow driven by randomly distributed recharge*, Advances in Water Resources 34.11, 2011, pp. 1450–1463, ISSN: 0309-1708, doi:https://doi.org/10.1016/j.advwatres.2011.07.003, https://www.sciencedirect.com/science/article/pii/S0309170811001321

[Pav16] **Pavlova**, A. et al.: *Geothermal heat in a heat pump use*, IOP Conference Series: Earth and Environmental Science 43, 2016, p. 012025, doi:10.1088/1755-1315/43/1/012025, https://doi.org/10.1088/1755-1315/43/1/012025

[Paz21] **Paz**, C. et al.: *Fsi modeling on the effect of artery-aneurysm thickness and coil embolization in patient cases*, Computer Methods and Programs in Biomedicine 206, 2021, p. 106148, ISSN: 0169-2607, doi:https://doi.org/10.1016/j.cmpb.2021.106148, https://www.sciencedirect.com/science/article/pii/S0169260721002224

[Pop20] **Pophillat**, W. et al.: *Analytical solutions for predicting thermal plumes of groundwater heat pump systems*, Renewable Energy 147, 2020, pp. 2696–2707, ISSN: 0960-1481, doi:https://doi.org/10.1016/j.renene.2018.07.148, https://www.sciencedirect.com/science/article/pii/S0960148118309455

[Rai19] **Raissi**, M.; **Perdikaris**, P.; **Karniadakis**, G.: *Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics 378, 2019, pp. 686–707, ISSN: 0021-9991, doi:https://doi.org/10.1016/j.jcp.2018.10.045, https://www.sciencedirect.com/science/article/pii/S0021999118307125

[Ric04] **Richter**, S.: *Entwicklung einer Methode zur integralen Beschreibung und Optimierung urbaner Energiesysteme*, PhD thesis, Universität Augsburg, 2004

[Ric19] **Ricchi**, A. et al.: *Multi-physics ensemble versus atmosphereocean coupled model simulations for a tropical-like cyclone in the mediterranean sea*, Atmosphere 10.4, 2019, doi:10.3390/atmos10040202, https://www.mdpi.com/2073-4433/10/4/202

[Rip99] **Rippa**, S.: *An algorithm for selecting a good value for the parameter c in radial basis function interpolation*, 11, 1999, pp. 193–210

[Ris19] **Risseeuw**, D.: *Fluid Structure Interaction Modelling of Flapping Wings: Development and Validation of a General Open-source Fluid Structure Interaction Method with Analysis of Flexible Flapping Wing Aerodynamics*, PhD thesis, Delft University of Technology, 2019, p. 190

[Ron15] **Ronneberger**, O.; **Philipp**, F.; **Thomas**, B.: *U-net: convolutional networks for biomedical image segmentation*, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Cham: Springer International Publishing, 2015, pp. 234–241

[Ruh19] **Ruhnau**, O.; **Hirth**, L.; **Praktiknjo**, A.: *Time series of heat demand and heat pump efficiency for energy system modeling.* Sci Data, 2019, doi:https://doi.org/10.1038/s41597-019-0199-y

[Sat07] **Sathe**, S. et al.: *Fluidstructure interaction modeling of complex parachute designs with the space time finite element techniques.* Computers Fluids 36.1, 2007, pp. 127–135

[Sch11] **Scheuerer**, M.: *An alternative procedure for selecting a good value for the parameter c in RBF-interpolation*, 2011, pp. 105–126, doi:10.1007/s10444-010-9146-3

[Sch15] **Scheufele**, K.: *Robust Quasi-Newton Methods for Partitioned Fluid-Structure Simulations*, MA thesis, Universität Stuttgart, 2015

[Sch18] **Scheufele**, K.: *Coupling Schemes and Inexact Newton for Multi-Physics and Coupled Optimization Problems*, PhD thesis, Universität Stuttgart, 2018

[Sch22] **Schmidt**, P. et al.: *Simulation of flow in deformable fractures using a quasi-Newton based partitioned coupling approach.* Comput Geosci 26, 2022, pp. 381–400

[Sel13] **Self**, S.; **Reddy**, B.; **Rosen**, M.: *Geothermal heat pump systems: status review and comparison with other heating options*, Applied Energy, 2013, doi:https://doi.org/10.1016/j.apenergy.2012.01.048

[Sha17] **Shankar**, V.: *The overlapped radial basis function-finite difference (rbf-fd) method: a generalization of rbf-fd*, Journal of Computational Physics 342, 2017, pp. 211–228

[Ska18] **Skansi**, S.: *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*, 1st ed., Springer Cham, 2018, doi:https://doi.org/10.1007/978-3-319-73004-2

[Sla13] **Slattery**, S. R.; **Wilson**, P. P. H.; **Pawlowski**, R. P.: *The data transfer kit: a geometric rendezvous-based tool for multiphysics data transfer*, 2013, https://www.osti.gov/biblio/22212795

[Söd17] **Söderberg**, R. et al.: *Toward a digital twin for real-time geometry assurance in individualized production*, CIRP Annals 66.1, 2017, pp. 137–140, ISSN: 0007-8506, doi:https://doi.org/10.1016/j.cirp.2017.04.038, https://www.sciencedirect.com/science/article/pii/S0007850617300380

[Sol05] **Solecki**, W. D. et al.: *Mitigation of the heat island effect in urban new jersey*, Global Environmental Change Part B: Environmental Hazards 6.1, 2005, pp. 39–49, ISSN: 1464-2867, doi:https://doi.org/10.1016/j.hazards.2004.12.002, https://www.sciencedirect.com/science/article/pii/S1464286705000045

[Spe20] **Spenke**, T.; **Hosters**, N.; **Behr**, M.: *A multi-vector interface quasi-Newton method with linear complexity for partitioned fluidstructure interaction*, Computer Methods in Applied Mechanics and Engineering 361, 2020, ISSN: 00457825, doi:10.1016/j.cma.2019.112810, arXiv: 2001.07947

[Tan20] **Tang**, M.; **Liu**, Y.; **Durlofsky**, L.: *A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems*, Journal of Computational Physics 413, 2020, p. 109456, doi:https://doi.org/10.1016/j.jcp.2020.109456

[Tan21] **Tang**, M.; **Liu**, Y.; **Durlofsky**, L. J.: *Deep-learning-based surrogate flow modeling and geological parameterization for data assimilation in 3d subsurface flow*, Computer Methods in Applied Mechanics and Engineering 376, 2021, p. 113636, ISSN: 0045-7825, doi:https://doi.org/10.1016/j.cma.2020.113636, https://www.sciencedirect.com/science/article/pii/S0045782520308215

[Ter20] **Terahara**, T. et al.: *Heart valve isogeometric sequentially-coupled FSI analysis with the spacetime topology change method*, Comput Mech 65, 2020, pp. 1167–1187, doi:https://doi.org/10.1007/s00466-019-01813-0

[Thu19] **Thuerey**, N. et al.: *Deep learning methods for reynolds-averaged navier-stokes simulations of airfoil flows.* American Institute of Aeronautics and Astronautics 58, 2019, pp. 686–707, doi:https://doi.org/10.2514/1.J058291

[Tre16] **Trehan**, S.: *Surrogate Modeling for Subsurface Flow: A New Reduced-Order Model and Error Estimation Procedures.* PhD thesis, Stanford University, 2016

[Udd14] **Uddin**, M.: *On the selection of a good value of shape parameter in solving time-dependent partial differential equations using rbf approximation method*, Applied Mathematical Modelling 38.1, 2014, pp. 135–144, ISSN: 0307-904X, doi:https://doi.org/10.1016/j.apm.2013.05.060, https://www.sciencedirect.com/science/article/pii/S0307904X13003831

[Uek16] **Uekermann**, B.: *Partitioned Fluid-Structure Interaction on Massively Parallel Systems*, PhD thesis, Technische Universität München, 2016

[Van09] **Van Brummelen**, E.: *Added mass effects of compressible and incompressible flows in fluid-structure interaction*, J. Appl. Mech. 76, 2009, p. 021206, doi:https://doi.org/10.1115/1.3059565

[Ver07] **Versteeg**, H.; **Malalasekera**, W.: *An introduction to computational fluid dynamics: The finite volume method*, 2nd ed., Prentice Hall, 2007

[Vie07] **Vierendeels**, J. et al.: *Implicit coupling of partitioned fluidstructure interaction problems with reduced order models*, Computers Structures 85.11, 2007, Fourth MIT Conference on Computational Fluid and Solid Mechanics, pp. 970–976, ISSN: 0045-7949, doi:https://doi.org/10.1016/j.compstruc.2006.11.006, https://www.sciencedirect.com/science/article/pii/S0045794906003865

[Vir20] **Virtanen**, P. et al.: *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods 17, 2020, pp. 261–272, doi:10.1038/s41592-019-0686-2

[Wal05] **Walhorn**, E. et al.: *Fluidstructure coupling within a monolithic model involving free surface flows.* Comput. Struct. 83, 2005, pp. 2100–2111, doi:https://doi.org/10.1016/j.compstruc.2005.03.010

[Wal11] **Walker**, H.; **Ni**, P.: *Anderson acceleration for fixed-point iterations*, SIAM Journal on Numerical Analysis 49, 2011, pp. 1715–1735

[Wan20] **Wang**, Y.; **Lin**, G.: *Efficient deep learning techniques for multiphase flow simulation in heterogeneous porousc media*, Journal of Computational Physics 401, 2020, p. 108968, ISSN: 0021-9991, doi:https://doi.org/10.1016/j.jcp.2019.108968, https://www.sciencedirect.com/science/article/pii/S0021999119306734

[Wel98] **Weller**, H.; **Tabor**, G.; **Jasak**, H.: *A tensorial approach to computational continuum mechanics using object-oriented techniques.* Comput. Phys. 1998

[Wol17] **Wolf**, K. et al.: *Mpcci: neutral interfaces for multiphysics simulations*, Scientific Computing and Algorithms in Industrial Simulations: Projects and Products of Fraunhofer SCAI, ed. by **Griebel**, M.; **Schüller**, A.; **Schweitzer**, M. A., Springer International Publishing, 2017, pp. 135–151, doi:10.1007/978-3-319-62458-7_7, https://doi.org/10.1007/978-3-319-62458-7_7

[Wri08] **Wriggers**, P.: *Nonlinear Finite Element Methods*, 1st ed., Springer Berlin, Heidelberg, 2008, doi:https://doi.org/10.1007/978-3-540-71001-1

[Yao15] **Yao**, G. et al.: *Implicit local radial basis function interpolations based on function values*, Applied Mathematics and Computation 265, 2015, pp. 91–102, doi:10.1016/j.amc.2015.04.107, http://dx.doi.org/10.1016/j.amc.2015.04.107

[Zhu19] **Zhu**, Y. et al.: *Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data*, Journal of Computational Physics 394, 2019, pp. 56–81, ISSN: 0021-9991, doi:https://doi.org/10.1016/j.jcp.2019.05.024, https://www.sciencedirect.com/science/article/pii/S0021999119303559

Declaration of Authorship

I hereby declare that this thesis titled:

Computational Methods for Partitioned Simulation Coupling:
Applications in Multi-Physics Simulation and Energy Infrastructure Optimization

was independently completed.

Information taken directly or indirectly from external sources is properly marked as such.

Stuttgart, 27 October, 2022