Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Invarianten von Ansatzfunktionen beim Basiswechsel

Kamigar Mohamadsharief

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. rer. nat. Miriam Schulte
Betreuer/in:	Dr. Stefan Zimmer
Beginn am:	8. Dezember 2021
Beendet am:	8. Juni 2022

Kurzfassung

Diese Bachelorarbeit beschäftigt sich mit der formalen Verifikation des newtonschen Ansatzes für die Interpolation. Hierfür wird ein Basiswechsel hergeleitet und bewiesen, dass die Transformationen der Basis die korrekten Ansatzfunktionen berechnen, wodurch die Verifikation des gesamten Algorithmus folgt. Es wird auch erörtert inwiefern sich Herangehensweisen für den Beweis und Umformungen des Algorithmus auf andere Algorithmen in der Numerik übertragen lassen.

Inhaltsverzeichnis

1 Einleitung

Wenn herausgefunden werden soll, ob ein Programm eine gewünschte Funktionalität erfüllt, wird üblicherweise das Programm auf verschiedene Eingaben überprüft. Diese Herangehensweise zeigt jedoch nur wie das Programm auf die vordefinierten Eingaben reagiert, aber beweist nicht dessen Korrektheit. Somit kann der Programmierer nur überzeugt sein, dass das Programm korrekt abläuft, was das Programm nicht unbedingt fehlerfrei macht und keinem formalen Beweis gleicht. Um die Korrektheit eines Programms zu beweisen eignet sich die Anwendung des Hoare-Kalküls, welches mithilfe von mathematischer Logik ein formales System beschreibt. Im Hoare-Kalkül werden für ein auszuführendes Programmsegment jeweils ein Prädikat für die Vorbedingung definiert, welches vor der Ausführung des Programmsegmentes gilt und ein Prädikat für die Nachbedingung, welches nach der Ausführung gilt. Die formale Korrektheit des Programmsegmentes ist gegeben wenn bei geltender Vorbedingung das Programmsegment ausgeführt wird und anschließend die Nachbedingung gilt. Im Allgemeinen wird für die Vorbedingung der initiale Zustand vor Ausführung des Programmsegments gewählt und die gewünschte Funktionalität beschreibt die Nachbedingung. Diese Herangehensweise wird für jede Anweisung im Programm angewandt, mit dem Ziel, dass das Programm als Ganzes formal verifiziert wird. Kommt in einem Programm eine Schleife vor, wird für die Verifikation der Schleife neben einer Vor- und Nachbedingung ein weiteres Prädikat benötigt, welches in der Schleife invariant ist. Invarianz bedeutet hier, dass das Prädikat sowohl vor- und nach der Schleife gilt als auch vor- und nach jedem Schleifendurchlauf. In LAFF-On Programming for Correctness [MG20] präsentieren M. E. Myers und R. A. van de Geijn das Hoare-Kalkül im Detail und erläutern die theoretischen Grundlagen, welche benötigt werden für die Beweisführung in der Verifikation von Programmsegmenten. Außerdem legen sie dar, wie durch gegebene Vor- und Nachbedingungen systematisch Hypothesen für Invarianten von Schleifen konstruiert werden können und darüber hinaus, wie die Anweisungen in der Schleife hergeleitet werden können. In der Forschung wird das Hoare-Kalkül beispielsweise in eingebetteten Softwaresystemen für die Verifikation angewandt von L. Zou et al. [ZZW+13] Das Werkzeug KeY [BH16] von R. Bubel und R. Hähnle ermöglicht eine automatische formale Verifikation von Algorithmen in Java, wobei nur die Eingabe von Schleifeninvarianten ein nichttrivialen Teil der Verifikation darstellt.

W. Gander[Gan05] stellt verschiedene Ansätze für die Polynominterpolation vor und untersucht die Basiswechsel zwischen den Ansatzfunktionen, welche durch unter anderem in der Lagrange-Basis und in der Newton-Basis repräsentiert werden. Die Bachelorarbeit vom Philip Urban [Urb21] beschäftigt sich mit der Verifikation von drei Algorithmen aus der Numerik. Unter anderem verifiziert er die Korrektheit der dividierten Differenzen, welche die führenden Koeffizienten der Basisfunktionen des newtonschen Algorithmus für die Interpolation sind.
In dieser Bachelorarbeit wird mithilfe des Hoare-Kalküls die formale Verifikation der Ansatzfunktionen des newtonschen Algorithmus für die Interpolation durchgeführt. Diese Ansatzfunktionen werden während der Berechnung der dividierten Differenzen durch einen Basiswechsel von der Lagrange-Basis in die Newton-Basis hinübergeführt. Um diesen Beweis durchführen zu können, werden im zweiten Kapitel die grundlegenden Regeln des Hoare-Kalküls erklärt. Daraufhin wird

durch Umformungen der Algorithmus verändert, sodass neben der Berechnung der dividierten Differenzen, die Ansatzfunktionen durch einen Wechsel in der Basis berechnet werden. Hierbei ist wichtig zu wissen, dass diese Umformungen das interpolierte Polynom nicht ändern zur Laufzeit des Algorithmus. Dadurch kann gezeigt werden, dass die Verifikation der Ansatzfunktionen in diesem Fall auch die Korrektheit der dividierten Differenzen impliziert. In Kapitel 4 werden Hypothesen für Schleifeninvarianten formuliert, welche systematisch hergeleitet werden können. Darauf aufbauend, wird dann die formale Verifikation durchgeführt. Zum Schluss findet noch eine Diskussion und ein Ausblick statt.

2 Grundlagen

Hoare-Tripel

Dieses Kapitel basiert auf den Grundlagen des Hoare-Kalküls wie sie in LAFF-On Programming for Correctness [MG20] vorgestellt sind.

Mithilfe des Hoare-Kalküls lässt sich formal die Korrektheit von Programmen beweisen. Für einzelne Programmsegmente wird die Notation des Hoare-Tripel verwendet, wobei ein Hoare-Tripel aus den Prädikaten Q, welches die Vorbedingung des Programmsegments S beschreibt und R für die Nachbedingung.

Das Hoare-Tripel

$$\{Q\}$$
$$S$$
$$\{R\}$$

evaluiert zu $TRUE$, wenn das Programmsegment S in einem Zustand ausgeführt wird, in welchem Q gilt, und nach der Ausführung im resultierenden Zustand R gilt. Außerdem muss die Ausführung von S in einer endlichen Zeit erfolgen. In dieser Arbeit werden nur Programmsegmente untersucht, welche mathematische Zuweisungen sind, daher wird die Terminierung hier angenommen. Gibt es keine Vorbedingung wird $TRUE$ als Vorbedingung gewählt.

Aus Platz- und Lesbarkeitsgründen wird $\{Q\}S\{R\}$ verwendet.

Komposition

Bei einer Komposition von Anweisungen $S_1; S_2$ lassen sich die zugehörigen Hoare-Tripel der Form
$\{Q\}S_1\{P\} \wedge \{P\}S_2\{R\} \Rightarrow \{Q\}S_1; S_2\{R\}$
zu einem Hoare-Tripel zusammensetzen.

Schwächste Vorbedingung

Um die Korrektheit der Hoare-Tripel zu zeigen, benutzt man die Definition der schwächsten Vorbedingung $wp(\text{"}S\text{"}, R)$, welche alle Zustände beschreibt, bei welchen nach der Ausführung von S ein Zustand existiert in welchem R gilt.
Das Hoare-Tripel $\{Q\}S\{R\}$ evaluiert genau dann zu $TRUE$, wenn $Q \Rightarrow wp(\text{"}S\text{"}, R)$.
Dies ist der Fall, da um das Hoare-Tripel zu validieren muss ein Zustand, welcher von der

schwächsten Vorbedingung beschrieben wird vorliegen. Also muss Q Zustände beschreiben, welche durch $wp(\text{"}S\text{"}, R)$ gedeckt werden, da sonst das Hoare-Tripel nicht validiert.

Schwächste Vorbedingung für Zuweisungen

Die Berechnung der schwächsten Vorbedingung für Zuweisungen findet wie folgt statt:

für $S : x := E$ wobei E ein mathematischer Ausdruck ist
$wp(\text{"}x := E\text{"}, R) = R^x_{(E)}$

Das Prädikat $R^x_{(E)}$ beschreibt das Prädikat R, bei welchem alle freien Vorkommnisse von x durch den Ausdruck (E) ersetzt werden. Diese Regel wird im weiteren Verlauf dieser Arbeit die Zuweisungsregel genannt.

Schwächste Vorbedingung für Kompositionen von Zuweisungen

Die schwächste Vorbedingung von Hoare-Tripeln mit n Anweisungen wird durch

$$\begin{aligned} wp(\text{"}S_1; S_2; ...; S_n\text{"}, R) &= wp(\text{"}S_1\text{"}, wp(\text{"}S_2; ...; S_n\text{"}, R)) \\ &= wp(\text{"}S_1\text{"}, wp(\text{"}S_2\text{"}, wp(..., wp(\text{"}S_n\text{"}, R)))) \end{aligned}$$

Diese Regel wird im weiteren Verlauf dieser Arbeit die Kompositionsregel für schwächste Vorbedingungen genannt.

WHILE Theorem

Das WHILE Theorem zeigt, wie die partielle Korrektheit von WHILE-Schleifen der Form:

while G **do**
  S
end while

bewiesen werden kann, wobei $S = S_1; ...; S_n$ aus n Anweisungen zusammengesetzt ist und G hier die Schleifenbedingung definiert.

Zunächst wird die WHILE-Schleife mit Prädikaten annotiert:

$\{Q\}$
$\{INV\}$
while G **do**
    $\{INV \wedge G\}$
    S
    $\{INV\}$
end while
$\{INV \wedge \neg G\}$
$\{R\}$

- INV ist hier die Schleifeninvariante, welche vor dem ersten Schleifendurchlauf gilt, weshalb $Q \Rightarrow INV$ gelten muss.
- Beim erstmaligem Betreten der WHILE-Schleife gilt die Schleifeninvariante INV, da keine Anweisungen ausgeführt wurden und nur die Schleifenbedingung G überprüft wurde und auch gilt.
- Wenn $\{INV \wedge G\}S\{INV\}$ gilt, dann gilt INV auch am Ende eines Schleifendurchlaufs und am Anfang des nächsten Schleifendurchlaufs.
- Wird die WHILE-Schleife verlassen so gilt INV immer noch und $\neg G$, da am Ende des letzten Schleifendurchlaufs INV galt und beim Verlassen der WHILE-Schleife keine Anweisungen ausgeführt wurden.
- Wenn schließlich $INV \wedge \neg G \Rightarrow R$ gilt ist die Korrektheit der WHILE-Schleife im Bezug zur Nachbedingung R bewiesen.

Für WHILE-Schleifen der obigen Form muss daher für die partielle Korrektheit folgendes bewiesen werden:

1. $Q \Rightarrow INV$
2. $\{INV \wedge G\}S\{INV\}$
3. $INV \wedge \neg G \Rightarrow R$

Für die vollständige Korrektheit neben der partiellen Korrektheit noch die Terminierung der WHILE-Schleife bewiesen werden. Beim Gebrauch des WHILE-Theorems werden hier ausschließlich WHILE-Schleifen behandelt, welche umgeformte FOR-Schleifen sind, die von einem endlichen Eingabeparameter abhängen.
Daher wird die Terminierung hier angenommen und für die vollständige Korrektheit genügt es hier die partielle Korrektheit zu beweisen.

$\{Q\}$**while**$(G)S$**endwhile**$\{R\}$ ist die Kurzschreibweise für ein Hoare-Tripel mit einer WHILE-Schleife als Programmsegment.

3 Herleitung des Basiswechsels

Im vorangegangenen Kapitel wurden die Grundlagen für das Hoare-Kalkül näher erläutert. Nun soll der newtonsche Algorithmus für die Interpolation so modifiziert werden, dass neben der Berechnung der Koeffizienten der Polynome in der Newton-Basis, auch die Polynome der Lagrange-Basis durch Transformationen in die Newton-Basis umgeformt werden. Dabei ist zu beachten, dass am Anfang des Algorithmus das zu interpolierende Polynom durch die Lagrange-Basis dargestellt wird. Außerdem soll sich bei der Berechnung der dividierten Differenzen und der Ansatzfunktionen in der Newton-Basis das zu interpolierende Polynom nicht verändern. Diese Eigenschaft wird in diesem Kapitel erneut aufgegriffen und ist maßgebend für die Verifikation des Algorithmus.

Dividierte Differenzen

Der newtonsche Algorithmus für die Interpolation in 3.1 erhält als Eingabe die paarweise verschiedenen Stützpunkte der zu interpolierenden Funktion und berechnet die führenden Koeffizienten $[x_i, ..., x_n]f$ der Polynome in der Newton-Basis. Diese Koeffizienten werden im newtonschen Ansatz dividierte Differenzen genannt.

Eingabe: $n + 1$ Stützpunkte $(x_i, f(x_i))$, $0 \leq i \leq n$

Algorithmus 3.1 Newtonsches Interpolationsverfahren

1: **for** $i = 0, ..., n$ **do**
2: $\quad d_i := f(x_i)$;
3: **end for**
4: **for** $k = 1, ..., n$ **do**
5: $\quad$ **for** $i = n, ..., k$ **do**
6: $\quad\quad d_i := (d_i - d_{i-1})/(x_i - x_{i-k})$;
7: $\quad$ **end for**
8: **end for**
9: **return** $[x_0]f = d_0, ..., [x_0, ..., x_n]f = d_n$

Das zu interpolierende Polynom ergibt sich aus:

$$P(x) = \sum_{i=0}^{n} \left(d_i \cdot \pi_i(x) \right) \text{ mit } \pi_i(x) = \prod_{k=0}^{i-1} (x - x_k) \text{ und}$$
$$\pi(x) = \begin{pmatrix} \pi_0(x) & \dots & \pi_n(x) \end{pmatrix}^\top$$

wobei hier $\pi(x)$ die Newton-Basis darstellt

Die Polynome in der Newton-Basis werden im modifizierten Algorithmus aus den Polynomen in der Lagrange-Basis berechnet, welche wie folgt definiert sind:

$$l_i(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k} \text{ mit } P(x) = \sum_{i=0}^{n} f_i \cdot l_i(x)$$
$$l(x) = \begin{pmatrix} l_0(x) & \dots & l_n(x) \end{pmatrix}^\top$$

außerdem wird $l(x)$ die Lagrange-Basis genannt.

Initalisierung

Die dividierten Differenzen sind bei der Initialisierung bereits gleich den Funktionswerten an den Stützstellen, weshalb zur Initialisierung die Ansatzfunktionen in der Lagrange-Basis repräsentiert werden können. φ_i sind die Ansatzfunktionen, welche bei der Initialisierung der Lagrange-Basis gleichen.
Die Initialisierung des newtonschen Algorithmus beim Basiswechsel:

for $i = 0, ..., n$ **do**
  $d_i := f(x_i)$;
  $\varphi_i(x) := l_i(x)$;
end for

Dann gilt:

$$P(x) = \sum_{i=0}^{n} d_i \cdot \varphi_i(x) = \sum_{i=0}^{n} f(x_i) \cdot l_i(x)$$

Schleifenkörper

Im Endzustand gilt

$$\forall i \in [0,n] : (d_i = [x_i, ..., x_n]f \wedge \varphi_i(x) = \pi_i(x)) \Rightarrow P(x) = \sum_{i=0}^{n} d_i \cdot \varphi_i(x)$$
$$= \sum_{i=0}^{n} [x_i, ..., x_n]f \cdot \pi_i(x)$$

Dies soll auch während jedem Schleifendurchlauf der inneren FOR-Schleife gelten, weil dadurch im gesamten Algorithmus hinweg das Polynom erhalten bleibt. Das hat zur Folge, dass am Ende des Algorithmus das interpolierte Polynom vorliegt und die Polynome der Newton-Basis. Wenn also gezeigt wird, dass am Ende die Polynome der Newton-Basis vorliegen und sich das Polynom während des Algorithmus nicht verändert, müssten die Koeffizienten der Newton-Basis, sprich die dividierten Differenzen korrekt berechnet worden sein, da sonst das zu interpolierende Polynom nicht korrekt dargestellt wird.

Für einen Schleifendurchlauf der inneren FOR-Schleife liegt am Anfang der Vektor d vor, welcher ein beliebiger sein kann aus $\mathbb{R}^{n+1}$, welcher hier als d^{alt} definiert wird. $\varphi_i(x)$ wird vor dem Schleifendurchlauf als $\varphi(x)^{alt}$ definiert. Nach dem Schleifendurchlauf liegen d^{neu}, φ^{neu} vor, an welchen eine Basistransformation durchgeführt wurde.
Dann muss gelten:

$$\sum_{a=0}^{n} d_a^{alt} \cdot \varphi_a^{alt}(x) \stackrel{!}{=} \sum_{a=0}^{n} d_a^{neu} \cdot \varphi_a^{neu}(x)$$

Für d^{alt} werden die Vektoren der kanonischen Basis $B = \{e_0, ..., e_n\}$ des Vektorraums $\mathbb{R}^{n+1}$ gewählt mit:

$$b \in [0,n] : d^{alt} = e_b$$

$\forall b \notin \{i-1, i\}$:

$$d^{alt} = d^{neu}$$

$$\sum_{a=0}^{n} d_a^{alt} \cdot \varphi_a^{alt}(x) = d_b^{alt} \cdot \varphi_b^{alt}(x)$$
$$= \varphi_b^{alt}(x)$$

Mit $d_b^{alt} = d_b^{neu}$ gilt: $\varphi_b^{alt}(x) = \varphi_b^{neu}(x)$

$b = i - 1$:

$$d_{i-1}^{neu} = d_{i-1}^{alt} = 1 \wedge d_i^{neu} = \frac{d_i^{alt} - d_{i-1}^{alt}}{x_i - x_{i-k}} = -\frac{1}{x_i - x_{i-k}}$$

$$\Rightarrow \varphi_{i-1}^{alt} = \varphi_{i-1}^{neu} - \frac{1}{x_i - x_{i-k}} \varphi_i^{neu}$$

Auflösen nach φ_i^{neu} fliefert:

$$\varphi_{i-1}^{neu} = \varphi_{i-1}^{alt} + \frac{1}{x_i - x_{i-k}} \varphi_i^{neu} \tag{3.1}$$

$b = i$:

$$d_i^{neu} = \frac{d_i^{alt} - d_{i-1}^{alt}}{x_i - x_{i-k}} = \frac{1}{x_i - x_{i-k}}$$
$$\Rightarrow \varphi_i^{alt} = \frac{d_i^{alt} - d_{i-1}^{alt}}{x_i - x_{i-k}} \varphi_i^{neu}$$

Auflösen nach φ_{i-1}^{alt} liefert:

$$\varphi_i^{neu} = (x_i - x_{i-k}) \cdot \varphi_i^{alt} \tag{3.2}$$

Durch einsetzen von 3.2 in 3.1 erhält die Gleichung die Form :

$$\begin{aligned} \varphi_{i-1}^{neu} &= \varphi_{i-1}^{alt} + \frac{1}{x_i - x_{i-k}} \varphi_i^{neu} \\ &= \varphi_{i-1}^{alt} + \frac{1}{x_i - x_{i-k}} (x_i - x_{i-k}) \cdot \varphi_i^{alt} \\ &= \varphi_{i-1}^{alt} + \varphi_i^{alt} \end{aligned}$$

Dadurch ergeben sich die Anweisungen:

$$\varphi_{i-1}(x) = \varphi_{i-1}(x) + \varphi_i(x)$$
$$\varphi_i(x) = (x_i - x_{i-k}) \cdot \varphi_i(x)$$

welche bei jedem Durchlauf der inneren FOR-Schleife ausgeführt werden.
Weil dadurch:

$$P(x) = \sum_{i=0}^{n} d_i \cdot \varphi_i(x)$$

bei jedem Schleifendurchlauf gilt, muss bewiesen werden, dass Während des Algorithmus auch tatsächlich die Ansatzfunktionen in der Newton-Basis berechnet werden.

Das newtonsche Algorithmus für die Interpolation mit einem Basiswechsel von der Lagrange-Basis in die Newton-Basis sieht dann wie folgt aus:

Algorithmus 3.2 Newtonscher Algorithmus für die Interpolation mit Basiswechsel

1: **for** $i = 0, ..., n$ **do**
2: $d_i := f(x_i)$
3: $\varphi_i(x) := l_i(x)$
4: **end for**
5: **for** $k = 1, ..., n$ **do**
6: **for** $i = n, ..., k$ **do**
7: $d_i := (d_i - d_{i-1})/(x_i - x_{i-k})$
8: $\varphi_{i-1}(x) = \varphi_{i-1}(x) + \varphi_i(x)$
9: $\varphi_i(x) = (x_i - x_{i-k}) \cdot \varphi_i(x)$
10: **end for**
11: **end for**
12: **return** $d_0, ..., d_n \wedge \varphi_0, ..., \varphi_n$

Matrixschreibweise

Um die Anweisungen für den Basiswechsel alternativ herzuleiten, eignet es sich den Algorithmus in Matrixschreibweise zu notieren.

Dann wird für den Vektor $\varphi(x) = \begin{pmatrix} \varphi_0(x) \\ \vdots \\ \varphi_n(x) \end{pmatrix}$

die Matrix mit der Form $\phi = \begin{pmatrix} \varphi_0(x_0) & \cdots\cdots & \varphi_0(x_n) \\ \vdots & & \vdots \\ \varphi_n(x_0) & \cdots\cdots & \varphi_n(x_n) \end{pmatrix}$ verwendet.

Die Funktionswerte an den Stützstellen $f = \begin{pmatrix} f_0 \\ \vdots \\ f_n \end{pmatrix}$ werden durch $f = \phi^\top d$ berechnet.

Das zu interpolierende Polynom lässt sich mit $P(x) = d^\top(\phi \cdot l(x)) = d^\top \pi(x)$ darstellen.

Die Anweisung:

7. $d_i = (d_i - d_{i-1})/(x_i - x_{i-k})$

wird durch die äquivalente Anweisung:

7. $d = A(i,k) \cdot d$

$$\text{mit} \quad A(i,k) = \begin{array}{c} \begin{matrix} & & i-1 & i & \\ \end{matrix} \\ \begin{pmatrix} 1 & \ddots & & & \\ & \ddots & \ddots & & \\ & & \frac{-1}{(x_i - x_{i-k})} & \frac{1}{(x_i - x_{i-k})} & \ddots \\ & & & \ddots & \ddots \\ & & & & 1 \end{pmatrix} \end{array} \begin{matrix} \\ \\ \\ \\ i \end{matrix}$$

ersetzt.

Um nun herzuleiten, welche Matrix X die Matrix ϕ transformieren soll, müssen die Funktionswerte der Stützstellen während dem Algorithmus erhalten bleiben, daher:

$$f = \phi^\top d = (X\phi)^\top \cdot A(i,k)d$$

Diese Gleichung geht für $X = A(i,k)^{-\top}$ auf, da:

$$\begin{aligned} f &= (A(i,k)^{-\top}\phi)^\top \cdot A(i,k)d \\ &= \phi^\top A(i,k)^{-1} \cdot A(i,k)d \\ &= \phi^\top d \end{aligned}$$

Also wird die Matrix ϕ mit $A(i,k)^{-\top}$ transformiert, welche wie folgt aussieht:

$$A(i,k)^{-\top} = \begin{array}{c} \begin{matrix} & & i-1 & i & \\ \end{matrix} \\ \begin{pmatrix} 1 & \ddots & & & \\ & \ddots & & & \\ & & 1 & 1 & \\ & & 0 & x_i - x_{i-k} & \ddots \\ & & & \ddots & \ddots \\ & & & & 1 \end{pmatrix} \end{array} \begin{matrix} \\ \\ \\ i-1 \\ i \end{matrix}$$

Der resultierende Algorithmus für newtonschen Ansatz für die Interpolation mit einem Basiswechsel muss so modifiziert werden, sodass keine FOR-Schleifen mehr darin vorkommen, da das Hoare-Kalkül nur mit WHILE-Schleifen operieren kann. Die FOR-Schleife für die Initialisierung kann wegfallen und durch Zuweisungen von Vektoren und Matrizen ersetzt werden. Der Vektor für die dividierten Differenzen wird dem Vektor der Funktionswerte der Stützstellen initialisiert. Die Matrix ϕ für die Ansatzfunktionen wird mit der Einheitsmatrix in $\mathbb{R}^{n+1}$ initialisiert, wegen:

$$\forall i, j \in [0, n] : \phi_{i,j} = l_i(x_j)$$
$$\text{und } \; l_i(x_j) = \begin{cases} 1 & i = j \\ 0 & \text{sonst} \end{cases}$$

Die weiteren FOR-Schleifen werden in WHILE-Schleifen umgeändert.

Algorithmus 3.3 Angepasstes newtonsches Interpolationsverfahren mit Basiswechsel in Matrixschreibweise

```
1:  k = 1
2:  i = n
3:  d = f
4:  φ = I_{n+1}
5:  while k ≤ n do
6:      i = n
7:      while i ≥ k do
8:          d = A(i, k) · d
9:          φ = A(i, k)^{-⊤} · φ
10:         i = i − 1
11:     end while
12:     k = k + 1
13: end while
14: return d, φ
```

Zum Schluss fehlt noch, den Algorithmus mit Hoare-Tripeln zu annotieren, welche vom WHILE Theorem abgeleitet werden:

Algorithmus 3.4 Angepasstes newtonsches Interpolationsverfahren mit Hoare-Tripeln

1: $k = 1$
2: $i = n$
3: $d = f$
4: $\phi = I_{n+1}$
5: $\{INV_{outer}\}$
6: **while** $k \leq n$ **do**
7: $\quad \{INV_{outer} \wedge (k \leq n)\}$
8: $\quad i = n$
9: $\quad \{INV_{inner}\}$
10: $\quad$ **while** $i \geq k$ **do**
11: $\quad\quad \{INV_{inner} \wedge (i \geq k)\}$
12: $\quad\quad d = A(i,k) \cdot d$
13: $\quad\quad \phi = A(i,k)^{-\top} \cdot \phi$
14: $\quad\quad i = i - 1$
15: $\quad\quad \{INV_{inner}\}$
16: $\quad$ **end while**
17: $\quad \{INV_{inner} \wedge \neg(i \geq k)\}$
18: $\quad k = k + 1$
19: $\quad \{INV_{outer}\}$
20: **end while**
21: $\{INV_{outer} \wedge \neg(k \leq n)\}$
22: **return** d, ϕ

4 Konstruktion der Schleifeninvarianten

Um systematisch Schleifeninvarianten für WHILE-Schleifen zu bestimmen, ist das Betrachten der Vor und Nachbedingungen wichtig, da sie den Start- und Endzustand darstellen und die Schleifeninvariante unter anderem diese beiden Zustände vor und nach der Schleife decken muss. Weil die Invariante nach jedem Durchlauf der Schleife gilt, muss diese auch den Fortschritt nach jedem Schleifendurchlauf festhalten. Die Schleifeninvariante soll nur so konstruiert sein, dass bei der Verifikation des Algorithmus bewiesen wird, dass die Ansatzfunktionen in der Newton-Basis berechnet werden.

Die folgende Herleitung der Schleifeninvarianten ist kein Beweis dafür, dass diese die richtige Invarianten beschreibt. Die Richtigkeit der Invarianten ist erst gegeben, wenn die Schleifen mithilfe der Invarianten unter der Anwendung des WHILE Theorems verifiziert sind.

4.1 Invariante der äußeren WHILE-Schleife

Um den den Zustand der Matrix ϕ in der Invariante festzuhalten, wird ein Prädikat benötigt, welches abhängig von der Schleifenvariable k diesen Zustand beschreibt. Das Prädikat wird $P(k)$ genannt. Wobei k auf den Umfang der Schleifenvariable beschränkt wird, mit $(k \leq n+1)$. Dass die Beschränkung des Umfangs der zulässigen Werte für die Prädikate nötig ist, wird im nächsten Kapitel beim Beweis des Algorithmus gezeigt.

$P(k)$ muss daher zum Zeitpunkt k den Zustand der einzelnen Zeilen der Matrix festhalten. Es ist sinnvoll $P(k)$ in Bereiche der Matrix ϕ aufzuteilen.
Die Zeilen, welche bereits dem Endzustand entsprechen und daher nicht mehr vom Algorithmus modifiziert werden, werden mit dem Prädikat $FIN(k)$ beschrieben. Die Zeilen, welche nicht dem Endzustand entsprechen und daher vom Algorithmus noch modifiziert werden, werden mit dem Prädikat $INTER(k)$ beschrieben.

Daraus lässt sich folgern, dass $P(k)$ sich aus

$$P(k) \equiv FIN(k) \wedge INTER(k)$$

zusammensetzt.

Um die Prädikate $FIN(k), INTER(k)$ zu bestimmen können die Vor- und Nachbedingungen aufschlussreich sein, welche gegeben sind mit: Vorbedingung der äußeren Schleife:

$$(k = 1) \wedge (i = n) \wedge (d = f), (\phi = I_{n+1}) \equiv (k = 1) \wedge (i = n) \wedge (d = f) \wedge P(1) \tag{4.1}$$

Nachbedingung der äußeren Schleife:

$$P(n+1) \equiv \forall i, j \in [0, n] : \phi_{i,j} = \prod_{d=0}^{i-1} (x_j - x_d) \tag{4.2}$$

Durch das Betrachten des Algorithmus kann man sehen, dass zum Zeitpunkt k die Zeilen im Intervall von $[0, k-1)$ nicht mehr modifiziert werden. Das bedeutet, dass diese Zeilen bereits den gewünschten Zustand der Nachbedingung erreicht haben müssen.
Somit lässt sich das Intervall in der Nachbedingung aufteilen in zwei Teile:

$$\left[\forall i^{'}, j \in [0, n] : \left(i^{'} \in [0, k-1) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1} (x_j - x_d)\right) \right.$$
$$\left. \wedge \left(i^{'} \in [k-1, n] \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1} (x_j - x_d)\right)\right] \wedge (k \leq n+1)$$

Wobei der erste Teil, Teil der Invariante ist.

Vorläufig kann festgehalten werden, dass $FIN(k)$ mindestens die Zeilen $[0, k-1)$ abdeckt mit:

$$FIN(k) \equiv \left[\forall i^{'}, j \in [0, n] : \left(i^{'} \in [0, k-1) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1} (x_j - x_d)\right)\right] \wedge (k \leq n+1)$$

Für diesen Algorithmus ist die Vorbedingung nicht sehr aufschlussreich, da die Matrix ϕ schon beim ersten Schleifendurchlauf alle Zeilen der Matrix modifiziert. Die einzige Information, welche die Vorbedingung für das Konstruieren der Invariante liefert, ist die Eigenschaft, dass zu beginn der Schleife $P(1)$ gelten muss. Es fehlt noch die Zeilen der Matrix ϕ im Intervall $[k-1, n]$ in die Invariante zu integrieren. Um diese in einer Invariante während jedem Schleifendurchlauf zu beschreiben ist es hilfreich die Entwicklung der Matrix ϕ an einem allgemeinem Beispiel zu analysieren.

Beispiel

Stützstellen $x_0, ..., x_3$ mit $f = \begin{pmatrix} f_0 \\ \vdots \\ f_3 \end{pmatrix}$

Zustand der Matrix ϕ für $k = 2$

$$\phi = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & x_1 - x_0 & x_1 - x_0 & x_1 - x_0 \\ 0 & 0 & x_2 - x_1 & x_2 - x_1 \\ 0 & 0 & 0 & x_3 - x_2 \end{pmatrix}$$

und $k = 3$

$$\phi = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ 0 & 0 & (x_2 - x_1)(x_2 - x_0) & (x_3 - x_1)(x_2 - x_0) \\ 0 & 0 & 0 & (x_3 - x_2)(x_3 - x_1) \end{pmatrix}$$

An diesem Beispiel lässt sich Erkennen, dass die Einträge der unteren Dreiecksmatrix also für $j < i'$ immer 0 sind. Begründen lässt sich dies durch die Nachbedingung, welche fordert, dass $\phi_{i',j} = \prod_{d=0}^{i'-1}(x_j - x_d)$. Weil für die untere Dreiecksmatrix $j < i'$ gilt, kommt ein Nullprodukt im Produkt vor. Die Einträge weichen auch während des Algorithmus nicht von 0 ab, weil eine Operation auf der Matrix nur eine Multiplikation ist und die andere Operation addiert zwei aufeinanderfolgende Zeilen, wobei $j < i'$ gilt und daher auch $j < i' + 1$.

Aufgrund dieser Observation, lässt sich die untere Dreiecksmatrix in den Endzustand eingliedern, weil dessen Einträge sich in der Matrix während der Schleifendurchläufe nicht ändern.

$FIN(k)$ lässt sich somit wie folgt definieren:

$$FIN(k) \equiv \left[\forall i', j \in [0, n] : \left(i' \in [0, k-1) \vee (j < i') \Rightarrow \phi_{i',j} = \prod_{d=0}^{i'-1} (x_j - x_d) \right) \right]$$

Aus dem Beispiel wird ersichtlich, bei dem Schleifendurchlauf die Zeile 1 zum letzten mal durch Operationen geändert wird und daher den Endzustand erreicht. Zeile 2 und 3 werden beim nächsten Schleifendurchlauf noch geändert und stellen hier den Zustand der Zeilen dar, welche noch nicht dem Endzustand entsprechen und daher noch geändert werden. Beim genaueren Betrachten fällt auf, dass die einzelnen Terme dem Produkt $\pi_{i',j} = \prod_{d=0}^{i'-1}(x_j - x_d)$ in absteigender Reihenfolge im Bezug auf den Index der Zeile i', mit jedem Schleifendurchlauf hinzugefügt werden, wobei noch $(x_i - x_{i-k})$ durch die Anweisung $\phi = A(i,k)^{-\top} \cdot \phi$ multipliziert wird.

Daher wird $INTER(k)$ wie folgt beschreiben:

$$INTER(k) \equiv \left[\forall i', j \in [0,n] : \left(i' \in [k-1,n] \wedge (j \geq i') \Rightarrow \phi_{i',j} = (x_{i'} - x_{i-k+1}) \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k+1}(x_j - x_d)}\right)\right] \wedge (k \leq n+1)$$

Weil bei der Initialisierung der Teiler zu 0 wird für die Einträge in der Matrix entlang der Diagonale, muss entlang dieser das Intervall aufgeteilt werden. Damit lässt sich $INTER(k)$ darstellen mit:

$$INTER(k) \equiv \left[\forall i', j \in [0,n] : \left(i' \in [k-1,n] \wedge (j = i') \Rightarrow \phi_{i',j} = \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k}(x_j - x_d)}\right) \wedge \left(i' \in [k-1,n] \wedge (j > i') \Rightarrow \phi_{i',j} = (x_{i'} - x_{i'-k+1}) \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k+1}(x_j - x_d)}\right)\right] \wedge (k \leq n+1)$$

Damit ergibt sich die Invariante der äußeren WHILE-Schleife:

$$INV_{outer} \equiv \left[\forall i', j \in [0,n] : \left(i' \in [0,k-1) \vee (j < i') \Rightarrow \phi_{i',j} = \prod_{d=0}^{i'-1}(x_j - x_d)\right) \wedge \left(i' \in [k-1,n] \wedge (j = i') \Rightarrow \phi_{i',j} = \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k}(x_j - x_d)}\right) \wedge \left(i' \in [k-1,n] \wedge (j > i') \Rightarrow \phi_{i',j} = (x_{i'} - x_{i'-k+1}) \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k+1}(x_j - x_d)}\right)\right] \wedge (k \leq n+1)$$

$\Longleftrightarrow$

$INV_{outer} \equiv P(k) \wedge (k \leq n+1)$

4.2 Invariante der inneren WHILE-Schleife

Um die Invariante der inneren WHILE-Schleife INV_{inner} herzuleiten muss der Zustand der Matrix ϕ an den Zeitpunkten k und i festgehalten werden.
Das Prädikat $P(i,k)$ beschreibt den Zustand der Matrix ϕ an den Zeitpunkten k und i. Das Prädikat

$P(k)$ kann in das Prädikat $P(i,k)$ für die Zeilen übernommen werden, die nicht von i abhängen. Das Prädikat $FIN(k)$ von $P(k)$ kann daher übernommen werden. Unter anderem fehlt es an dem Prädikat noch den Zustand der Zeilen in der Matrix festzuhalten, welche von der inneren WHILE-Schleife bereits verändert wurden und nicht mehr zum Zeitpunkt k verändert werden. Es sind die Zeilen im Intervall von $[i+1,n]$, dessen Einträge in der Matrix ϕ den Zustand für $k+1$ darstellen. Diese Kenntnis wird aus der Nachbedingung der inneren WHILE-Schleife gewonnen, welche wie folgt definiert ist:

$$P(k+1) \wedge (k \leq n) \tag{4.3}$$

Daher gilt für die Zeilen $[i+1,n]$ in ϕ:

$$\left[\forall i^{'}, j \in [0,n] : \left(i^{'} \in (i,n] \wedge (j \geq i^{'}) \Rightarrow \phi_{i^{'},j} = (x_i - x_{i-k}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}\right)\right]$$
$$\wedge (i \geq k-1) \wedge (k \leq n)$$

Die Zeilen im Intervall von $[k-1, i-1]$ können von $INTER(k)$ aus $P(k)$ übernommen werden, da diese von der inneren WHILE-Schleife zum Zeitpunkt k noch nicht verändert wurden und daher den Zustand zum Zeitpunkt k annehmen.

Es fehlt nur noch die Zeile i zu betrachten, welche den Zustand für $k+1$ darstellt bevor $(x_i - x_{i-k})$ im nächsten Schleifendurchlauf multipliziert wird und lässt sich daher so darstellen:

$$\left[\forall i^{'}, j \in [0,n] \left(i^{'} = i \Rightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}\right)\right]$$
$$\wedge (i \geq k-1) \wedge (k \leq n)$$

Somit lässt sich die Invariante der inneren WHILE-Schleife INV_{inner} wie folgt definieren, wobei die Zeile i in den zweiten Teilausdruck zusammengefasst ist:

$$\begin{aligned} INV_{inner} \equiv \Bigg[& \forall i^{'}, j \in [0,n] : \left((i^{'} \in [0,k-1) \vee (j < i^{'})) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d) = \pi_{i^{'}}(x_j)\right) \\ & \wedge \left((i^{'} \in [k-1,i-1] \wedge (j = i^{'})) \vee (i^{'} = i) \Rightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}\right) \\ & \wedge \left(i^{'} \in [k-1,i-1] \wedge (j > i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k+1}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}\right) \\ & \wedge \left(i^{'} \in (i,n] \wedge (j \geq i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}\right)\Bigg] \\ & \wedge (i \geq k-1) \wedge (k \leq n) \end{aligned}$$

$$\Longleftrightarrow INV_{inner} \equiv P(i,k) \wedge (i \geq k-1) \wedge (k \leq n)$$

Für $P(k)$ wird von nun an $P(n,k)$ verwendet mit $i = n$ und damit wird:

$$INV_{outer} = P(n,k) \wedge (k \leq n+1)$$

Jetzt muss noch geprüft werden, ob in der Initialisierung und im Endzustand des Algorithmus, $P(i,k)$ die Matrix ϕ zu den Zeitpunkten i, k korrekt repräsentiert.

Initialisierung der äußeren WHILE-Schleife:

$$\begin{aligned}
P(n,1) \equiv & \forall i', j \in [0,n] : \left(i' \in [0,0) \vee (j < i') \Rightarrow \phi_{i',j} = \prod_{d=0}^{i'-1} (x_j - x_d) = \pi_{i'}(x_j) \right) \\
& \wedge \left((i' \in [0,n-1] \wedge (j = i')) \vee (i' = n) \Rightarrow \phi_{i',j} = \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-1}(x_j - x_d)} = 1 \right) \\
& \wedge \left(i' \in [0,n-1] \wedge (j > i') \Rightarrow \phi_{i',j} = (x_{i'} - x_{i'}) \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'}(x_j - x_d)} = 0 \right) \\
& \wedge \left(i' \in (n,n] \wedge (j \geq i') \Rightarrow \phi_{i',j} = (x_{i'} - x_{i'-1}) \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-1}(x_j - x_d)} \right) \\
\equiv & \forall i', j \in [0,n] : \left((j < i') \vee (j > i') \Rightarrow \phi_{i',j} = 0 \right) \\
& \wedge \left((j = i') \Rightarrow \phi_{i',j} = 1 \right) \\
\equiv & \phi = I_{n+1}
\end{aligned} \tag{4.4}$$

Endzustand der äußeren WHILE-Schleife:

$$
\begin{aligned}
P(n,n+1) \equiv & \forall i^{'}, j \in [0,n] : \left((^{'} \in [0,n) \vee (j < i^{'}) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1} (x_j - x_d) = \pi_{i^{'}}(x_j) \right) \\
& \wedge \left((i^{'} \in [n,n-1] \wedge (j = i^{'})) \vee (i^{'} = n) \Rightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-n-1}(x_j - x_d)} \right) \\
& \wedge \left(i^{'} \in [n,n-1] \wedge (j > i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-n}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-n}(x_j - x_d)} \right) \\
& \wedge \left(i^{'} \in (n,n] \wedge (j \geq i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-n-1}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-n-1}(x_j - x_d)} \right) \\
\equiv & \forall i^{'}, j \in [0,n] : \left(i^{'} \in [0,n) \vee (j < i^{'}) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1} (x_j - x_d) = \pi_{i^{'}}(x_j) \right) \\
& \wedge \left((i^{'} = n) \wedge (j = i^{'}) \Rightarrow \phi_{n,n} = \frac{\prod_{d=0}^{n-1}(x_n - x_d)}{\prod_{d=0}^{-1}(x_n - x_d)} = \prod_{d=0}^{n-1}(x_n - x_d) \right) \\
\equiv & \forall i^{'}, j \in [0,n] : \left(i^{'} \in [0,n] \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1} (x_j - x_d) \right)
\end{aligned}
\tag{4.5}
$$

Es folgt also, dass für die Initialisierung und den Endzustand der Zustand der Matrix ϕ durch $P(i,k)$ korrekt beschrieben wird.

Es ist noch hilfreich für die Verifikation der inneren WHILE-Schleife, dessen Endzustand darzustellen mit:

$$
\begin{aligned}
P(n,k+1) \equiv & \forall i^{'}, j \in [0,n] : \left((i^{'} \in [0,k) \vee (j < i^{'})) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1} (x_j - x_d) = \pi_{i^{'}}(x_j) \right) \\
& \wedge \left((i^{'} \in [k,n-1] \wedge (j = i^{'})) \vee (i^{'} = n) \Rightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k-1}(x_j - x_d)} \right) \\
& \wedge \left(i^{'} \in [k,n-1] \wedge (j > i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)} \right) \\
& \wedge \left(i^{'} \in (n,n] \wedge (j \geq i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-n-1}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-n-1}(x_j - x_d)} \right) \\
\equiv & \forall i^{'}, j \in [0,n] : \left((i^{'} \in [0,k) \vee (j < i^{'})) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1} (x_j - x_d) = \pi_{i^{'}}(x_j) \right) \\
& \wedge \left((i^{'} \in [k,n-1] \wedge (j = i^{'})) \vee (i^{'} = n) \Rightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k-1}(x_j - x_d)} \right) \\
& \wedge \left(i^{'} \in [k,n-1] \wedge (j > i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)} \right)
\end{aligned}
\tag{4.6}
$$

Ob aber die Invarianten korrekt sind, welche momentan nur Hypothesen sind, wird sich erst beim Beweis des Algorithmus herausstellen, welcher im kommenden Abschnitt behandelt wird.

5 Verifikation des Algorithmus

5.1 Korrektheitsbeweis der äußeren WHILE-Schleife

Im vorherigen Abschnitt wurden Hypothesen von Invarianten für die äußere und die innere WHILE-Schleife konstruiert sind korrekte Invarianten, wenn der Algorithmus mithilfe dieser Invarianten verifiziert wird. Im folgenden Unterkapitel wird zunächst unter der Annahme, dass die innere WHILE-Schleife das korrekte Ergebnis liefert und damit verifiziert ist, die äußere WHILE-Schleife verifiziert. Der Grund dafür ist, die Lesbarkeit zu vereinfachen und weil der Beweis für die Verifikation der inneren WHILE-Schleife technisch anspruchsvoller ist und ein eigenes Unterkapitel benötigt.
Kommentare bei Rechnungen werden mit einem "// "gekennzeichnet.

Zunächst wird die Vor- und Nachbedingung der äußeren WHILE-Schleife definiert.

Vorbedingung aus 4.1:

$$(k = 1) \wedge (i = n) \wedge (d = f) \wedge P(n, 1)$$

Diese Vorbedingung muss vor der Schleife gelten und das Hoare-Tripel:

$$\{TRUE\}k = 1; i = n; d = f; \phi = I_{n+1}\{(k = 1) \wedge (i = n) \wedge (d = f) \wedge P(n, 1)\}$$

beschreibt die Initialisierung des Algorithmus.

Mithilfe der schwächsten Vorbedingung lässt sich das Hoare-Tripel als korrekt beweisen, indem:

$$\begin{aligned}
TRUE &\Rightarrow wp(\text{"}k = 1; i = n; d = f; \phi = I_{n+1}\text{"}, (k = 1) \wedge (i = n) \wedge (d = f) \wedge P(n, 1)) \\
&\quad \text{// aus der Kompositionsregel für schwächste Vorbedingungen folgt:} \\
&\quad \Rightarrow wp(\text{"}k = 1\text{"}, wp(\text{"}i = n\text{"}, wp(\text{"}d = f\text{"}, wp(\text{"}\phi = I_{n+1}\text{"}, (k = 1) \wedge (i = n) \wedge (d = f) \wedge P(n, 1))))) \\
&\quad \text{// aus 4.4 folgt: } P(n, 1) \equiv \phi = I_{n+1} \\
&\quad \text{// aus der Zuweisungsregel folgt:} \\
&\quad \Rightarrow wp(\text{"}k = 1\text{"}, wp(\text{"}i = n\text{"}, wp(\text{"}d = f\text{"}, (k = 1) \wedge (i = n) \wedge (d = f) \wedge (I_{n+1} = I_{n+1})))) \\
&\quad \Rightarrow wp(\text{"}k = 1\text{"}, wp(\text{"}i = n\text{"}, (k = 1) \wedge (i = n) \wedge (f = f) \wedge (I_{n+1} = I_{n+1}))) \\
&\quad \Rightarrow wp(\text{"}k = 1\text{"}, (k = 1) \wedge (n = n) \wedge (f = f) \wedge (I_{n+1} = I_{n+1})) \\
&\quad \Rightarrow (1 = 1) \wedge (n = n) \wedge (f = f) \wedge (I_{n+1} = I_{n+1}) \\
&\quad \Rightarrow TRUE
\end{aligned}$$

Somit gilt die Vorbedingung.

Die Nachbedingung aus 4.2 lautet:

$$P(n, n+1)$$

Um die Korrektheit der äußeren WHILE-Schleife zu Beweisen, wird das WHILE Theorem angewandt und liefert die Schritte für die zu erbringenden Beweise, welche die folgenden sind:

1. $(k = 1) \wedge (i = n) \wedge (d = f) \wedge P(n, 1) \Rightarrow INV_{outer} = P(n, k) \wedge (k \leq n + 1)$
2. $\{INV_{outer} \wedge (k \leq n)\} i = n; \textbf{inner}; k = k + 1 \{INV_{outer}\}$

   Mit **inner** anstelle der ausgeschriebenen inneren WHILE-Schleife aufgrund von Platz- und Lesbarkeitsgründen in diesem Unterkapitel.
3. $INV_{outer} \wedge \neg(k \leq n) \Rightarrow P(n, n+1)$

Hier soll 1. die Verifikation der Initialisierung der WHILE-Schleife, 2. die Verifiktation des Schleifenkörpers und 3. der Abschluss der WHILE-Schleife sein.

Korrektheit der Initialisierung:

Die Invariante soll vor der Schleife gelten, daher die Implikation:

$$(k = 1) \wedge (i = n) \wedge (d = f) \wedge P(n, 1) \Rightarrow INV_{outer} = P(n, k) \wedge (k \leq n + 1)$$

Beweis:

$$\begin{aligned}
(k = 1) \wedge (i = n) \wedge (d = f) \wedge P(n, 1) &\Rightarrow (k = 1) \wedge (i = n) \wedge P(n, 1) \\
&\text{// Einsetzen von } (k = 1) \text{ in } P(n, 1) \\
&\Rightarrow (k = 1) \wedge (i = n) \wedge P(n, k) \\
&\text{// wegen: } (k = 1) \wedge (n \geq 0) \Rightarrow (k \leq n + 1) \\
&\Rightarrow (k = 1) \wedge (i = n) \wedge P(n, k) \wedge (k \leq n + 1) \\
&\Rightarrow P(n, k) \wedge (k \leq n + 1) \\
&\Rightarrow INV_{outer}
\end{aligned}$$

Damit gilt die Invariante INV_{outer} vor der Schleife.

Korrektheit des Schleifenkörpers:

Zu beweisen ist das Hoare-Tripel:

$$\{INV_{outer} \wedge (k \leq n)\} i = n; \textbf{inner}; k = k + 1 \{INV_{outer}\}$$

welches zu $TRUE$ evaluiert, wenn bei geltender Invariante INV_{outer} und Schleifenbedingung $(k \leq n)$ nach einem Schleifendurchlauf die Invariante INV_{outer} gilt.

Zunächst lässt sich das Hoare-Tripel nach der Kompositionsregel aufteilen in die folgenden Hoare-Tripel:

$$\{INV_{outer} \wedge (k \leq n)\} i = n, \textbf{inner}; k = k + 1 \{INV_{outer}\} \Leftrightarrow$$
$$\{INV_{outer} \wedge (k \leq n)\} i = n \{INV_{outer} \wedge (k \leq n) \wedge (i = n)\} \tag{5.1}$$
$$\wedge \{INV_{outer} \wedge (k \leq n) \wedge (i = n)\} \textbf{inner} \{P(n, k+1) \wedge (k \leq n)\} \tag{5.2}$$
$$\wedge \{P(n, k+1) \wedge (k \leq n)\} k = k + 1 \{INV_{outer}\} \tag{5.3}$$

Wenn die Hoare-Tripel 5.1 - 5.3 bewiesen werden, dann folgt daraus, dass das Zusammengesetzte Hoare-Tripel bewiesen ist. Um das Hoare-Tripel 5.2 zu beweisen, muss das WHILE Theorem angewandt und die damit zusammenhängende Verifikation der inneren WHILE-Schleife gezeigt werden. Dies wird erst im nächsten Unterkapitel behandelt, daher wird für die Verifikation der äußeren WHILE-Schleife angenommen, dass dieses Hoare-Tripel zu $TRUE$ evaluiert. Das nächste Unterkapitel bestätigt mit der Verifikation der inneren WHILE-Schleife die Annahme und damit dann den Algorithmus im Ganzen. Also müssen hier noch 5.1 und 5.3 gezeigt werden.

Beweis für 5.1:

Aus dem Satz der schwächsten Vorbedingung evaluiert das Hoare-Tripel zu $TRUE$, wenn:

$$\begin{aligned} INV_{outer} \wedge (k \leq n) &\Rightarrow wp(\text{"}i = n\text{"}, INV_{outer} \wedge (k \leq n) \wedge (i = n)) \\ &\text{// aus der Zuweisungsregel folgt:} \\ &\Rightarrow INV_{outer} \wedge (k \leq n) \wedge (n = n) \\ &\Rightarrow INV_{outer} \wedge (k \leq n) \wedge TRUE \\ &\Rightarrow INV_{outer} \wedge (k \leq n) \end{aligned}$$

Es bleibt noch der Beweis für 5.3 aus:

$$\begin{aligned} P(n, k+1) \wedge (k \leq n) &\Rightarrow wp(\text{"}k = k + 1\text{"}, INV_{outer}) \\ &\Rightarrow wp(\text{"}k = k + 1\text{"}, P(n, k) \wedge (k \leq n + 1)) \\ &\text{// aus der Zuweisungsregel folgt:} \\ &\Rightarrow P(n, k+1) \wedge (k + 1 \leq n + 1) \\ &\Rightarrow P(n, k+1) \wedge (k \leq n) \end{aligned}$$

Damit gilt unter der Annahme, dass die innere WHILE-Schleife verifiziert ist, die Invariante INV_{outer} vor- und nach jedem Schleifendurchlauf.

Korrektheit des Abschlusses:

Nun muss gezeigt werden, dass nach der äußeren WHILE-Schleife, die Matrix ϕ für die Ansatzfunktionen in der Newton-Basis korrekt berechnet worden ist.

Beweis:

$$\begin{aligned} INV_{outer} \wedge \neg(k \leq n) &\Rightarrow P(n,k) \wedge (k \leq n+1) \wedge \neg(k \leq n) \\ &\Rightarrow P(n,k) \wedge (k \leq n+1) \wedge (k > n) \\ &\Rightarrow P(n,k) \wedge (k = n+1) \\ &\text{// einsetzen von } (k = n+1) \\ &\Rightarrow P(n,n+1) \wedge (k = n+1) \\ &\Rightarrow P(n,n+1) \end{aligned}$$

Damit gilt:

$$INV_{outer} \wedge \neg(k \leq n) \Rightarrow P(n,n+1)$$

Aus 4.2 folgt, dass $P(n,n+1)$ den Endzustand beschreibt, in dem die Ansatzfunktionen in der Newton-Basis repräsentiert sind.

Die im letzten Kapitel angesprochene Bedeutung für das Einschränken von k mit $(k \leq n+1)$ ist hier deutlich gemacht um auf die Form $(k = n+1)$ zu kommen.
Generell soll für Invarianten von Schleifen, die freien Variablen, welche in Prädikaten als Eingabe vorkommen, auf ihre zulässigen Eingaben beschränkt werden. Grund dafür ist, dass Korrektheitsbeweise von WHILE-Schleifen im Abschluss generell wie eben vorgestellt strukturiert sind.

Da dies das Ende des Algorithmus darstellt ist der Algorithmus verifiziert worden, unter der Annahme, dass die innere WHILE-Schleife verifiziert ist.

5.2 Korrektheitsbeweis der inneren WHILE-Schleife

Im vorangegangenen Kapitel wurde die Verifikation des gesamten Algorithmus durchgeführt bis auf das Hoare-Tripel 5.2, welches wie folgt definiert ist:

$$\{INV_{outer} \wedge (k \leq n) \wedge (i = n)\}\textbf{while}(i \geq k)d = A(i,k)d; \phi = A(i,k)^{-\top}; i = i-1\textbf{endwhile}\{P(n,k+1) \wedge (k \leq n)\}$$

Wobei $INV_{outer} \wedge (k \leq n) \wedge (i = n)$ die Vorbedingung und $P(n,k+1) \wedge (k \leq n)$ die Nachbedingung der inneren WHILE-Schleife beschreiben.
Durch den Beweis dieses Hoare-Tripels wird der gesamte Algorithmus verifiziert, da dies der einzige Beweisschritt ist, dessen Richtigkeit angenommen wurde und noch fehlt.

Das WHILE Theorem liefert parallel zu 5.1 drei Beweisschritte, welche durchgeführt werden müssen um die innere WHILE-Schleife zu verifizieren:

1. $INV_{outer} \wedge (k \leq n) \wedge (i = n) \Rightarrow INV_{inner}$
2. $\{INV_{inner} \wedge (i \geq k)\}\textbf{while}(i \geq k)d = A(i,k)d; \phi = A(i,k)^{-\top}; i = i-1\textbf{endwhile}\{INV_{inner}\}$
3. $INV_{inner} \wedge \neg(i \geq k) \Rightarrow P(n,k+1) \wedge (k \leq n)$

Korrektheit der Initialisierung:

$$\begin{aligned} INV_{outer} \wedge (k \leq n) \wedge (i = n) &\Rightarrow P(n,k) \wedge (k \leq n+1) \wedge (k \leq n) \wedge (i = n) \\ &\text{// durch einsetzen von } (i = n) \text{ und } (i = n) \wedge (k \leq n+1) \Rightarrow (i \geq k-1) \\ &\text{// durch } (k \leq n) \wedge (k \leq n+1) \Rightarrow (k \leq n) \\ &\Rightarrow P(i,k) \wedge (i \geq k-1) \wedge (k \leq n) \wedge (i = n) \\ &\Rightarrow P(i,k) \wedge (i \geq k-1) \wedge (k \leq n) \\ &\Rightarrow INV_{inner} \end{aligned}$$

Also gilt INV_{inner} vor dem ersten Schleifendurchlauf der inneren WHILE-Schleife und laut dem WHILE Theorem gilt beim erstmaligen Betreten der WHILE-Schleife zusätzlich noch die Schleifenbedingung, sodass:

$$INV_{inner} \wedge (i \geq k)$$

gilt.

Korrektheit des Schleifenkörpers:

Das Hoare-Tripel:

$$\{INV_{inner} \wedge (i \geq k)\}\textbf{while}(i \geq k)d = A(i,k)d; \phi = A(i,k)^{-\top}; i = i - 1\textbf{endwhile}\{INV_{inner}\}$$

soll bewiesen werden.

Nach dem Satz für die schwächste Vorbedingung ist dieses Hoare-Tripel äquivalent zu:

$$\begin{aligned} INV_{inner} \wedge (i \geq k) &\Rightarrow wp(\text{"}d = A(i,k)d; \phi = A(i,k)^{-\top}; i = i - 1\text{"}, INV_{inner}) \\ &\quad \text{// aus der Kompositionsregel für schwächste Vorbedingungen folgt:} \\ &\Rightarrow \overbrace{wp(\text{"}d = A(i,k)d, \underbrace{wp(\phi = A(i,k)^{-\top}, \underbrace{wp(i = i - 1\text{"}, INV_{inner})}_{\textbf{a}})}_{\textbf{b}})}^{\textbf{c}} \end{aligned}$$

Diese schwächste Vorbedingung wird von **a**-**c** rekursiv evaluiert, sodass bei bei der Berechnung von **c** das Ergebnis der schwächsten Vorbedingung aller Anweisungen der inneren WHILE-Schleife steht.

schwächste Vorbedingung a:

$$\begin{aligned} wp(\text{"}i = i - 1\text{"}, INV_{inner}) &= wp(\text{"}i = i - 1\text{"}, P(i,k) \wedge (i \geq k - 1) \wedge (k \leq n)) \\ &\quad \text{// nach dem Satz der schwächsten Vorbedingung folgt:} \\ &= P(i-1,k) \wedge (i - 1 \geq k - 1) \wedge (k \leq n) \qquad (5.4) \\ &= P(i-1,k) \wedge (i \geq k) \wedge (k \leq n) \qquad (5.5) \end{aligned}$$

schwächste Vorbedingung b:

$$wp(\phi = A(i,k)^{-\top}\phi, wp(\text{"}i = i{-}1\text{"}, INV_{inner})) \overset{5.4}{=} wp(\phi = A(i,k)^{-\top}\phi, P(i{-}1,k) \wedge (i{-}1 \geq k{-}1) \wedge (k \leq n))$$

Da das Prädikat $P(i-1,k)$ die Einträge in der Matrix ϕ beschreibt, werden sie ersetzt durch das Ergebnis der Matrizenmultiplikation von $\phi = A(i,k)^{-\top}$:

$$\forall i', j \in [0,n] : \phi_{i',j} = \sum_{d=0}^{n} A^{-\top}_{i',d}(i,k) \cdot \phi_{d,j}$$

$P(i-1,k) \wedge (i \geq k) \wedge (k \leq n)$ nimmt nach der obigen Ersetzung folgende Form an:

$$= \forall i^{'}, j \in [0,n] : \left(i^{'} \in [0,k-1) \vee (j < i^{'}) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d) = \pi_{i^{'}}(x_j) \right) \tag{5.6}$$

$$\wedge \left(((i^{'} \in [k-1, i-2] \wedge (j = i^{'})) \vee (i^{'} = i-1)) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)} \right) \tag{5.7}$$

$$\wedge \left((i^{'} \in [k-1, i-2] \wedge (j > i^{'})) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = (x_{i^{'}} - x_{i^{'}-k+1}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)} \right) \tag{5.8}$$

$$\wedge \left(i^{'} \in (i-1, n] \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = (x_{i^{'}} - x_{i^{'}-k}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)} \right) \tag{5.9}$$

$$\wedge (i \geq k) \wedge (k \leq n)$$

Um die oben-stehenden Summen zu berechnen, ist es Hilfreich eine Fallunterscheidung der verschiedenen Werte durchzuführen, die die Einträge der Matrix $A(i,k)^{-\top}$ besitzen können:

$$\forall i^{'} \in [0,n] : A^{-\top}_{i^{'},d}(i,k) = \begin{cases} x_{i^{'}} - x_{i^{'}-k} & \text{für } (i = i^{'}) \wedge (d = i) \\ 1 & \text{für } ((i^{'} = d) \wedge (d \neq i)) \vee ((i^{'} = i-1) \wedge (i = d)) \\ 0 & \text{sonst} \end{cases} \tag{5.10}$$

Jetzt können die Teilausdrücke 5.6-5.9 jeweils so aufgeteilt werden, sodass die Summe $\sum_{d=0}^{n} A^{-T}_{i^{'},d}(i,k) \cdot \phi_{d,j}$ nur die drei folgenden Ergebnisse hat:

$$\forall i^{'}, j \in [0,n] : \left(i^{'} \notin \{i-1, i\} \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} \right)$$

$$\wedge \left(i^{'} = i-1 \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \phi_{i-1,j} + \phi_{i,j} = \phi_{i^{'},j} + \phi_{i^{'}+1,j} \right)$$

$$\wedge \left(i^{'} = i \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = (x_i - x_{i-k})\phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k})\phi_{i^{'},j} \right)$$

Dies folgt aus der Überlegung in 5.10, dass die Zeilen in der Matrix $A(i,k)^{-\top}$, welche nicht $\{i-1, i\}$ sind, die Zeilen einer Einheitsmatrix darstellen und für die Zeilen $\{i-1, i\}$ jeweils eine Transformation der Basispolynome durchgeführt wird.

Nun werden die Summen für die Teilausdrücke jeweils berechnet und am werden dann nach $\phi_{i^{'},j}$ aufgelöst, Zusammengefasst und vereinfacht:

Teilausdruck 5.6

Für den Fall $i^{'} \in [0, k-1)$ ist $i^{'} \notin \{i-1, i\}$, da $(i \geq k)$ gilt, also:

$$\forall i^{'}, j \in [0,n] : i^{'} \in [0,k-1) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d)$$

Für $j < i^{'}$ muss eine Fallunterscheidung zwischen $(j > i^{'})$ und $(j = i^{'})$ gemacht werden:

$$\forall i^{'}, j \in [0,n] : (j < i^{'}) \wedge (i^{'} \notin \{i-1, i\}) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d)$$

$$\forall i^{'}, j \in [0,n] : (j < i^{'}) \wedge (i^{'} = i-1) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \quad \phi_{i^{'},j} + \phi_{i^{'}+1,j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d)$$

$$\Leftrightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d) - \phi_{i^{'}+1,j}$$

// aus $(j < i^{'}+1) \Rightarrow \phi_{i^{'}+1,j} = \prod_{d=0}^{i^{'}}(x_j - x_d) = 0$ folgt:

$$\Leftrightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d)$$

$$\forall i^{'}, j \in [0,n] : (j < i^{'}) \wedge (i^{'} = i) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = (x_{i^{'}-1} - x_{i^{'}-k})\phi_{i^{'},j} = \prod_{d=0}^{i^{'}}(x_j - x_d)$$

$$\Leftrightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{(x_{i^{'}} - x_{i^{'}-k})}$$

// aus $(j < i^{'}+1) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d) = 0$ folgt:

$$\Leftrightarrow \phi_{i^{'},j} = 0$$

$$\Leftrightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d)$$

Somit lässt sich zusammenfassen:

$$\forall i^{'}, j \in [0,n] : \left(i^{'} \in [0,k-1) \vee (j < i^{'}) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d) = \pi_{i^{'}}(x_j) \right)$$

Teilausdruck 5.7

Für $i^{'} \in [k-1, i-2] \wedge (j = i^{'})$ ist $i^{'} \notin \{i-1, i\}$, da $(i \geq k)$ gilt, also:

$$\forall i^{'}, j \in [0,n] : i^{'} \in [k-1, i-2] \wedge (j = i^{'}) \Rightarrow \sum_{d=0}^{n} A_{i^{'},d}^{-\top}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}$$

Für $(i^{'} = i-1)$ muss auch eine Fallunterscheidung zwischen $(j > i^{'})$ und $(j = i^{'})$ gemacht werden:

$$\forall i^{'}, j \in [0,n] : (i^{'} = i-1) \wedge (j = i^{'}) \Rightarrow \sum_{d=0}^{n} A_{i^{'},d}^{-\top}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} + \phi_{i^{'}+1,j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}$$

$$\Leftrightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)} - \phi_{i^{'}+1,j}$$

// aus $(j = i^{'}) \Rightarrow (j < i^{'}+1) \Rightarrow \phi_{i^{'}+1,j} = \prod_{d=0}^{i^{'}}(x_j - x_d) = 0$ folgt:

$$\Leftrightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}$$

$$\forall i^{'}, j \in [0,n] : (i^{'} = i-1) \wedge (j > i^{'}) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} + \phi_{i^{'}+1,j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}$$

$$\Leftrightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)} - \phi_{i^{'}+1,j}$$

// aus $(j > i^{'}) \wedge (i^{'}+1 = i) \Rightarrow \phi_{i^{'}+1,j} = \frac{\prod_{d=0}^{i^{'}}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}$ folgt:

$$\Leftrightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)} - \frac{\prod_{d=0}^{i^{'}}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}$$

// Ersten Bruch um $(x_j - x_{i^{'}-k+1})$ erweitern:

// $(x_j - x_{i^{'}})$ aus $\prod_{d=0}^{i^{'}}(x_j - x_d)$ ausklammern:

$$\Leftrightarrow \phi_{i^{'},j} = \frac{\left(\prod_{d=0}^{i^{'}-1}(x_j - x_d)\right) \cdot (x_j - x_{i^{'}-k+1})}{\left(\prod_{d=0}^{i^{'}-k}(x_j - x_d)\right) \cdot (x_j - x_{i^{'}-k+1})} - \frac{\left(\prod_{d=0}^{i^{'}-1}(x_j - x_d)\right) \cdot (x_j - x_{i^{'}})}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}$$

$$\Leftrightarrow \phi_{i^{'},j} = \frac{\left(\prod_{d=0}^{i^{'}-1}(x_j - x_d)\right) \cdot (x_j - x_{i^{'}-k+1})}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)} - \frac{\left(\prod_{d=0}^{i^{'}-1}(x_j - x_d)\right) \cdot (x_j - x_{i^{'}})}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}$$

// Durch ausklammern von $\left(x_j - x_{i^{'}-k+1} - (x_j - x_{i^{'}})\right)$ ergibt sich:

$$\Leftrightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)} \cdot \left(x_j - x_{i^{'}-k+1} - (x_j - x_{i^{'}})\right)$$

$$\Leftrightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k+1}) \cdot \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}$$

Die Ergebnisse für den zweiten Teilausdruck fassen sich wie folgt zusammen:

$$\forall i^{'}, j \in [0,n] : \left(i^{'} \in [k-1, i-1] \wedge (j = i^{'}) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}\right)$$
$$\left((i^{'} = i-1) \wedge (j = i^{'}) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k+1}) \cdot \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}\right)$$

Teilausdruck 5.8

Für $i^{'} \in [k-1, i-2] \wedge (j > i^{'})$ ist $i^{'} \notin \{i-1, i\}$, da $(i \geq k)$ gilt, also:

$$\forall i^{'}, j \in [0,n] : i^{'} \in [k-1, i-2] \wedge (j > i^{'}) \Rightarrow \sum_{d=0}^{n} A^{-\top}_{i^{'},d}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k+1}) \cdot \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}$$

Teilausdruck 5.9

Für $i^{'} \in (i,n]$ ist $i^{'} \notin \{i-1,i\}$, also:

$$\forall i^{'}, j \in [0,n] : i^{'} \in (i,n] \wedge (j \geq i^{'}) \Rightarrow \sum_{d=0}^{n} A_{i^{'},d}^{-\top}(i,k) \cdot \phi_{d,j} = \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k}) \cdot \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}$$

Für $(i^{'} = i) \wedge (j \geq i^{'})$ gilt:

$$\forall i^{'}, j \in [0,n] : (i^{'} = i) \wedge (j \geq i^{'}) \Rightarrow \sum_{d=0}^{n} A_{i^{'},d}^{-\top}(i,k) \cdot \phi_{d,j} = (x_{i^{'}} - x_{i^{'}-k}) \cdot \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k}) \cdot \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}$$

$$\Leftrightarrow \phi_{i^{'},j} = \frac{(x_{i^{'}} - x_{i^{'}-k})}{(x_{i^{'}} - x_{i^{'}-k})} \cdot \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}$$

$$\Leftrightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}$$

Wenn die Ergebnisse der Teilausdrücke 5.6-5.9 zusammengefasst werden,
lässt sich $wp(\text{"}\phi = A(i,k)^{-\top}\phi\text{"}, P(i-1) \wedge (i \geq k) \wedge (k \leq n))$ wie folgt darstellen:

$$\begin{aligned}
&\left[\forall i^{'}, j \in [0,n] : \left(i^{'} \in [0,k-1) \vee (j < i^{'}) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d) = \pi_{i^{'}}(x_j)\right)\right. \\
&\wedge \left(i^{'} \in [k-1,i-1] \wedge (j = i^{'}) \Rightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}\right) \\
&\wedge \left((i^{'} = i-1) \wedge (j > i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k+1}) \cdot \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}\right) \\
&\wedge \left(i^{'} \in [k-1,i-2] \wedge (j > i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k+1}) \cdot \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}\right) \\
&\wedge \left(i^{'} \in (i,n] \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k}) \cdot \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}\right) \\
&\wedge \left.\left((i^{'} = i) \wedge (j \geq i^{'}) \Rightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}\right)\right] \\
&\wedge (i \geq k) \wedge (k \leq n)
\end{aligned}$$

Nach dem Vereinfachen folgt:

$$\left[\forall i^{'}, j \in [0,n] : \left(i^{'} \in [0,k-1) \vee (j < i^{'}) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1}(x_j - x_d) = \pi_{i^{'}}(x_j)\right)\right.$$
$$\wedge \left((i^{'} \in [k-1, i-1] \wedge (j = i^{'})) \vee (i^{'} = i) \Rightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}\right)$$
$$\wedge \left(i^{'} \in [k-1, i-1] \wedge (j > i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k+1}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k+1}(x_j - x_d)}\right)$$
$$\left.\wedge \left(i^{'} \in (i,n] \wedge (j \geq i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)}\right)\right]$$
$$\wedge (i \geq k) \wedge (k \leq n)$$

Das Prädikat in den eckigen Klammern ist genau das Prädikat $P(i,k)$, also gilt:

$$wp(\text{"}\phi = A(i,k)^{-\top}\phi\text{"}, P(i-1,k) \wedge (i \geq k) \wedge (k \leq n)) = P(i,k) \wedge (i \geq k) \wedge (k \leq n) \quad (5.11)$$

schwächste Vorbedingung c:

$wp(\text{"}d = A(i,k)d, wp(\phi = A(i,k)^{-\top}, wp(i = i{-}1\text{"}, INV_{inner}))) \overset{5.11}{=} wp(\text{"}d = A(i,k)d\text{"}, P(i,k) \wedge (i \geq k) \wedge (k \leq n))$

Weil für die Zuweisung $d = A(i,k)d$ kein d frei in $P(i,k) \wedge (i \geq k) \wedge (k \leq n)$ vorkommt, ist:

$$wp(\text{"}d = A(i,k)d\text{"}, P(i,k) \wedge (i \geq k) \wedge (k \leq n)) = P(i,k) \wedge (i \geq k) \wedge (k \leq n)$$

Somit muss nur noch gezeigt werden, dass $INV_{inner} \wedge (i \geq k) \Rightarrow P(i,k) \wedge (i \geq k) \wedge (k \leq n)$:

$$\begin{aligned} INV_{inner} \wedge (i \geq k) &\Rightarrow P(i,k) \wedge (i \geq k-1) \wedge (k \leq n) \wedge (i \geq k) \\ &\text{// wegen } (i \geq k-1) \wedge (i \geq k) \Rightarrow (i \geq k) \text{ gilt:} \\ &\Rightarrow P(i,k) \wedge (i \geq k) \wedge (k \leq n) \end{aligned}$$

Damit evaluiert das Hoare-Tripel zu $TRUE$ und die Schleifeninvariante gilt vor- und nach dem Ausführen der inneren WHILE-Schleife. Es bleibt nur noch zu zeigen, dass beim Verlassen der inneren WHILE-Schleife der Zustand der Matrix ϕ für die Schleifenvariable $k + 1$ gilt.

Korrektheit des Abschlusses

Schließlich muss noch gezeigt werden, dass:

$$INV_{inner} \wedge \neg(i \geq k) \Rightarrow P(n, k+1) \wedge (k \leq n)$$

$$\begin{aligned} INV_{inner} \wedge \neg(i \geq k) &\Rightarrow P(i,k) \wedge (i \geq k-1) \wedge (k \leq n) \wedge \neg(i \geq k) \\ &\Rightarrow P(i,k) \wedge (i \geq k-1) \wedge (k \leq n) \wedge (i < k) \\ &\text{// aus } (i \geq k-1) \wedge (i < k) \Rightarrow (i = k-1) \text{ folgt:} \\ &\Rightarrow P(i,k) \wedge (i = k-1) \wedge (k \leq n) \\ &\text{// einsetzen von } (i = k-1) \text{ in } P(i,k) \\ &\Rightarrow P(k-1,k) \wedge (k \leq n) \end{aligned}$$

Für $P(k-1,k) \wedge (k \leq n)$ ergibt sich:

$$\begin{aligned} &\forall i', j \in [0,n] : \left[\left((i' \in [0,k-1) \vee (j < i')) \Rightarrow \phi_{i',j} = \prod_{d=0}^{i'-1} (x_j - x_d) = \pi_{i'}(x_j) \right) \right. \\ &\wedge \left(((i' \in [k,k-2] \wedge (j = i')) \vee (i' = k-1)) \Rightarrow \phi_{i',j} = \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k}(x_j - x_d)} \right) \\ &\wedge \left((i' \in [k,k-2] \wedge (j > i')) \Rightarrow \phi_{i',j} = (x_{i'} - x_{i'-k+1}) \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k+1}(x_j - x_d)} \right) \\ &\left. \wedge \left(i' \in (k-1,n] \wedge (j \geq i') \Rightarrow \phi_{i',j} = (x_{i'} - x_{i'-k}) \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k}(x_j - x_d)} \right) \right] \\ &\wedge (k \leq n) \end{aligned}$$

Die Intervalle für das i' im zweiten und dritten Teilausdruck sind leer, daher stimmt die Implikation und die Teilausdrücke können wegfallen, bis auf die Bedingung $(i' = k-1)$, welche durch Folgende Überlegung in den ersten Teilausdruck absorbiert werden kann:

$$\forall i', j \in [0,n] : i' = k-1 \Rightarrow \phi_{i',j} \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k}(x_j - x_d)} = \prod_{d=0}^{i'-1}(x_j - x_d) = \pi_{i'}(x_j)$$

Der vierte Teilausdruck wird in die Fälle $j = i'$ und $j > i'$ unterteilt. Damit folgt:

$$\begin{aligned} &\forall i', j \in [0,n] : \left[\left((i' \in [0,k) \vee (j < i')) \Rightarrow \phi_{i',j} = \prod_{d=0}^{i'-1} (x_j - x_d) = \pi_{i'}(x_j) \right) \right. \\ &\left. \wedge \left(i' \in [k,n] \wedge (j = i') \Rightarrow \phi_{i',j} = \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k-1}(x_j - x_d)} \right) \right] \\ &\left[\wedge \left(i' \in [k,n] \wedge (j > i') \Rightarrow \phi_{i',j} = (x_{i'} - x_{i'-k}) \frac{\prod_{d=0}^{i'-1}(x_j - x_d)}{\prod_{d=0}^{i'-k}(x_j - x_d)} \right) \right] \\ &\wedge (k \leq n) \end{aligned}$$

Weil es keine Einträge in der Matrix für $(i^{'} = n) \wedge (j > i^{'})$ gibt, kann das Intervall vom dritten Teilausdruck auf $[k, n-1]$ verkürzt werden und der zweite Teilausdruck in einer anderen Form notiert werden:

$$
\begin{aligned}
&\equiv \forall i^{'}, j \in [0,n] : \Bigg[\left((i^{'} \in [0,k) \vee (j < i^{'})) \Rightarrow \phi_{i^{'},j} = \prod_{d=0}^{i^{'}-1} (x_j - x_d) = \pi_{i^{'}}(x_j) \right) \\
&\wedge \left((i^{'} \in [k, n-1] \wedge (j = i^{'})) \vee (i^{'} = n) \Rightarrow \phi_{i^{'},j} = \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k-1}(x_j - x_d)} \right) \\
&\wedge \left(i^{'} \in [k, n-1] \wedge (j > i^{'}) \Rightarrow \phi_{i^{'},j} = (x_{i^{'}} - x_{i^{'}-k}) \frac{\prod_{d=0}^{i^{'}-1}(x_j - x_d)}{\prod_{d=0}^{i^{'}-k}(x_j - x_d)} \right) \Bigg] \\
&\wedge (k \leq n)
\end{aligned}
$$

Aus 4.6 folgt, dass das Prädikat in den eckigen Klammern das Prädikat $P(n, k+1)$ darstellt, also:

$$INV_{inner} \wedge \neg(i \geq k) \Rightarrow P(n, k+1) \wedge (k \leq n)$$

Damit erfüllen sich alle Anforderungen des WHILE Theorems für die Verifikation der inneren WHILE-Schleife. Da im vorangegangenen Kapitel bereits gezeigt wurde, dass der Algorithmus verifiziert ist unter der Annahme, die innere WHILE-Schleife sei verifiziert, der Algorithmus nun vollständig verifiziert. Dazu gehört auch, dass die dividierten Differenzen verifiziert sind, was aus der Überlegung in Kapitel 3 folgt, weil sich das zu interpolierende Polynom nicht verändert.

6 Diskussion und Ausblick

In dieser Bachelorarbeit wurde der newtonsche Ansatz für die Interpolation verifiziert, wobei sich der Beweis nur auf die Verifikation der Ansatzfunktionen beschränkt. Trotzdem wird damit die Richtigkeit der dividierten Differenzen bewiesen, was eine andere Herangehensweise für die Verifikation des newtonschen Ansatzes für die Interpolation ist als Philip Urban mit seiner Bachelorarbeit [Urb21], dessen Beweis sich auf die dividierten Differenzen bezogen hat.
Der nächste Schritt wäre es, weitere Algorithmen zur Interpolation mit den vorgestellten Techniken zu verifizieren und herauszufinden, inwiefern sich die einzelnen Überlegungen dieser Bachelorarbeit auf diese Algorithmen übertragen.
Für die Herleitung des Basiswechsels ist es einfach die Basistransformationen zu bestimmen, indem die Matrizenschreibweise gewählt wird und nur das Inverse einer Matrix berechnet wird. Die Konstruktion der Schleifeninvarianten ist sehr abhängig von den ausgeführten Anweisungen aber es ist immer Hilfreich sich die Vor- und Nachbedingungen zu berücksichtigen, da diese in irgendeiner Form in der Invariante auftreten müssen. Außerdem benötigen die Invarianten Einschränkungen auf den Umfang der in ihr vorkommenden Variablen, was in dieser Bachelorarbeit ersichtlich wurde.
Die Korrektheitsbeweise lassen sich bis auf die Hoare-Tripel gut auf andere Algorithmen übertragen, da die Hoare-Tripel in den WHILE-Schleifen stark vom Algorithmus abhängen. An diesem Beweis wird deutlich, dass kurze Algorithmen aus der Numerik sich durchaus für einen Korrektheitsbeweis eignen, anstelle von einem herkömmlichen Test, welcher kein formaler Beweis ist.

Literaturverzeichnis

[BH16] R. Bubel, R. Hähnle. „KeY-Hoare“. In: *Deductive Software Verification – The KeY Book: From Theory to Practice*. Hrsg. von W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, P. H. Schmitt, M. Ulbrich. Cham: Springer International Publishing, 2016, S. 571–589 (zitiert auf S. 7).

[Gan05] W. Gander. „Change of basis in polynomial interpolation“. In: *Numerical Linear Algebra with Applications* 12 (2005) (zitiert auf S. 7).

[MG20] M. E. Myers, R. A. van de Geijn. *LAFF-On Programming for Correctness*. 2020 (zitiert auf S. 7, 9).

[Urb21] P. Urban. „Formale Verifikation von Basistransformationen für Funktionsdarstellungen“. In: (2021) (zitiert auf S. 7, 43).

[ZZW+13] L. Zou, N. Zhany, S. Wang, M. Fränzle, S. Qin. „Verifying simulink diagrams via a hybrid hoare logic prover“. In: *2013 Proceedings of the International Conference on Embedded Software (EMSOFT)*. IEEE. 2013, S. 1–10 (zitiert auf S. 7).

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift