

# **Implementierung finiter Elemente beliebiger Ordnung in 1D und Simulationen einer ebenen elektromagnetischen Welle mit dem Newmark-beta-Verfahren**

**BACHELORARBEIT**

zur Erlangung des akademischen Grades

**Bachelor of Science**

im Rahmen des Studiums

**Elektrotechnik und Informationstechnik**

eingereicht von

**Christian Maderthaner**

Matrikelnummer 11711298

an der Fakultät für Elektrotechnik und Informationstechnik

der Technischen Universität Wien

Betreuung: Dipl.-Ing. Dr.techn. Karl Hollaus

Wien, 23. November 2021

---

Christian Maderthaner

Karl Hollaus



# **Erklärung zur Verfassung der Arbeit**

Christian Maderthaner

Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, 23. November 2021

---

Christian Maderthaner



# Kurzfassung

Die Elektrodynamik, welche auch die Ausbreitung elektromagnetischer Wellen in einem allgemeinen Medium beschreibt, ist durch vier partielle Differentialgleichungen, den sogenannten Maxwell-Gleichungen und den Materialbeziehungen beschrieben.

Da analytische Lösungen nur für relativ einfache Problemstellungen angegeben werden können, ist die Verwendung eines numerischen Berechnungsverfahrens für komplizierte Geometrien der Materialien unumgänglich.

In dieser Bachelorarbeit ist ein 1D-Problemgebiet beschrieben, bei welchem eine ebene elektromagnetische Welle mit der Kreisfrequenz  $\omega$  von links in das Problemgebiet einläuft und beim Durchqueren mit dem Medium wechselwirkt. Dabei können die Materialparameter der Permittivität  $\varepsilon(x)$  und der elektrischen Leitfähigkeit  $\sigma(x)$  beliebig vorgegeben werden.

Das Problemgebiet wird in finite Elemente unterteilt. Als Basisfunktionen wurden Lagrange- und Legendre-Polynome beliebiger Ordnung implementiert. Gelöst wird das Problem mittels der Finiten-Elemente-Methode (FEM) und dem Newmark-beta-Verfahren als Zeitschrittverfahren.

Eine nichtreflektierende Randbedingung wurde verwendet um ein unbeschränktes Gebiet zu simulieren.

Nachdem das Problem und die numerischen Methoden zur Lösung beschrieben wurden, wird die numerische Näherungslösung  $u_h(x, t)$  mit der analytischen Lösung  $u(x, t)$  für spezielle Probleme gegenübergestellt.

Die Methoden zur Simulation wurden in Python Version 3.9.0 implementiert und sind im Anhang zu finden.



# Abstract

Electrodynamics, which also describes the propagation of electromagnetic waves in a general medium, is described by four partial differential equations, the so-called Maxwell equations along with the material relations.

Since analytical solutions are available only for relatively simple problems, the use of a numerical computation method is inevitable for complex geometries of the materials.

In this bachelor thesis a 1D problem is described, where a plane electromagnetic wave with the angular frequency  $\omega$  enters the problem at the left boundary and interacts with the medium while crossing it. The material parameters of the permittivity  $\varepsilon(x)$  and the electric conductivity  $\sigma(x)$  can be chosen arbitrarily.

The problem domain is subdivided into finite elements. Lagrange and Legendre polynomials of arbitrary order have been implemented as basis functions. The discrete problem is solved using the finite element method (FEM) and the Newmark-beta method as a time-stepping method.

A non-reflective boundary condition has been used to simulate an unbounded domain.

After describing the problem and discussing the numerical methods, the numerical solution  $u_h(x, t)$  is compared with the analytical solution  $u(x, t)$  for specific problems.

The methods have been implemented in Python, version 3.9.0, and can be found in the appendix.



# Inhaltsverzeichnis

<b>Kurzfassung</b>	v
<b>Abstract</b>	vii
<b>Inhaltsverzeichnis</b>	ix
<b>1 Einleitung</b>	1
<b>2 Problembeschreibung</b>	3
2.1 Wellengleichung . . . . .	4
2.2 Randbedingungen . . . . .	4
2.3 Anfangsbedingungen . . . . .	4
<b>3 Herleitung der dreidimensionalen Wellengleichung</b>	5
<b>4 Eindimensionale Wellengleichung</b>	7
<b>5 Randbedingungen</b>	9
5.1 Rechter Rand - nichtreflektierend . . . . .	9
5.2 Linker Rand - nichtreflektierend . . . . .	10
<b>6 Finite-Elemente-Methode - lineare Basisfunktionen</b>	13
6.1 Die schwache Formulierung . . . . .	13
6.2 Finite Elemente - Räumliche Diskretisierung . . . . .	15
6.3 Näherungslösung für lineare Basisfunktionen . . . . .	16
6.4 Numerische Integration . . . . .	19
<b>7 Höhere Ordnungen der Basisfunktionen</b>	27
7.1 Lagrange Polynombasis . . . . .	28
7.2 Hierarchische Polynombasis . . . . .	30
7.3 Weitere Berechnung . . . . .	32
<b>8 Lösung des gew. Differentialgleichungssystems - zeitliche Diskretisierung</b>	33
8.1 Newmark-beta-Verfahren . . . . .	33

8.2	Algorithmus zum Lösen eines Tridiagonalsystems . . . . .	35
<b>9</b>	<b>Vergleich der Näherungslösung mit der analytischen Lösung</b>	<b>37</b>
9.1	Ausbreitung im Vakuum . . . . .	37
9.2	Reflexion an einem perfekten Leiter . . . . .	45
9.3	Übergang der Welle zwischen zwei Medien unterschiedlicher Permittivität	50
9.4	Weitere Berechnungen . . . . .	53
<b>10</b>	<b>Python Source-Code</b>	<b>57</b>
10.1	test.py . . . . .	57
10.2	settings_domain1D.py . . . . .	60
10.3	main_FEM.py . . . . .	60
10.4	integration.py . . . . .	63
10.5	timeIntegrationMethod.py . . . . .	66
10.6	shapeFunctions.py . . . . .	68
10.7	plotResult.py . . . . .	70
	<b>Literaturverzeichnis</b>	<b>73</b>

# Einleitung

## ***Kapitel 2: Problembeschreibung***

In dem einleitenden Kapitel Problembeschreibung wird ein kurzer Überblick über das zu lösende Problem gegeben, ohne auf die numerische Berechnung einzugehen. Es wird dabei die Wellengleichung mit den Anfangs- und Randbedingungen vorgestellt.

## ***Kapitel 3: Herleitung der dreidimensionalen Wellengleichung***

Hier wird die dreidimensionale Wellengleichung aus den vier Maxwell-Gleichungen und den Verknüpfungsbeziehungen zwischen Flussdichte und Feldstärke hergeleitet. Dabei werden inhomogene Materialparameter der Permittivität  $\epsilon(\mathbf{r})$ , Permeabilität  $\mu(\mathbf{r})$  und elektrischen Leitfähigkeit  $\sigma(\mathbf{r})$  angesetzt.

## ***Kapitel 4: Eindimensionale Wellengleichung***

Ausgangspunkt für die eindimensionale Wellengleichung ist die im Kapitel 3 hergeleitete dreidimensionale Wellengleichung. Die Gleichung für den eindimensionalen Fall wird daraus abgeleitet.

## ***Kapitel 5: Randbedingungen***

Es werden in diesem Kapitel Randbedingungen beschrieben, sodass eine ebene Welle von links in das Problemgebiet einläuft und Wellenanteile am Rand nichtreflektiert aus dem Gebiet hinauslaufen können.

## **Kapitel 6: Finite-Elemente-Methode - lineare Basisfunktionen**

Nach Beschreibung und Herleitung der zu lösenden Differentialgleichung mit den beiden Randbedingungen wird in diesem Kapitel die FEM für eine lineare Approximation beschrieben. Dabei wird die schwache Formulierung hergeleitet, die finiten Elemente definiert, die Näherungslösung angesetzt, auf die numerische Integration der Matrixeinträge eingegangen und schlussendlich ein im Ort diskretisiertes gewöhnliches Differentialgleichungssystem angegeben.

## **Kapitel 7: Höhere Ordnungen der Basisfunktionen**

Nachdem im Kapitel 6 die Berechnung mit linearen Basisfunktionen beschrieben ist, wird in diesem Kapitel auf das Wesentliche eingegangen, um die Berechnung mit Polynombasen höherer Ordnung durchzuführen. Dabei wird die Lagrange-Polynombasis und die hierarchische Polynombasis mit Legendre-Polynomen definiert.

## **Kapitel 8: Lösung des gew. Differentialgleichungssystems - zeitliche Diskretisierung**

Um das erhaltene gewöhnliche Differentialgleichungssystem aus Kapitel 6 zeitlich zu diskretisieren, wird in diesem Kapitel das Newmark-beta-Verfahren vorgestellt. Weiters wird ein Algorithmus definiert, der für lineare Basisfunktionen zum effizienteren Lösen des Tridiagonalsystems dient.

## **Kapitel 9: Vergleich der Näherungslösung mit der analytischen Lösung**

Es werden Problemstellungen gewählt für die die analytischen Lösungen bekannt sind. Nach der numerischen Berechnung wird die Näherungslösung  $u_h(x, t)$  mit der analytischen Lösung  $u(x, t)$  verglichen. Dabei werden Simulationsparameter wie die Anzahl der finiten Elemente (FE), die Ordnung der Basisfunktionen und die Anzahl der Zeitschritte variiert. Um diesen Einfluss beurteilen zu können, wird der Fehler der Berechnung für das 1D-Problemgebiet grafisch dargestellt.

## **Kapitel 10: Python Source-Code**

In diesem Kapitel ist der für die numerische Simulation (FEM, Zeitschrittverfahren) verwendete Source-Code zu finden. Es wird kurz beschrieben, wie eine Simulation durch Ausführen des Moduls test.py durchgeführt werden kann.

# 2

## KAPITEL

## Problembeschreibung

Das behandelte Problem stellt eine sich in x-Richtung ausbreitende, harmonisch angelegte Welle dar, beschrieben durch die elektrische Feldstärke  $\mathbf{E}(\mathbf{r}, t)$  in einem linearen, inhomogenen, isotropen, verlustbehafteten und quellenfreien Medium, welches durch die Permittivität  $\varepsilon(\mathbf{r})$ , Permeabilität  $\mu_0$  und die elektrische Leitfähigkeit  $\sigma(\mathbf{r})$  charakterisiert ist.

Weiters handelt es sich bei dem Problem um ein in y- und z-Richtung unendlich ausgedehntes Gebiet mit konstanten Materialeigenschaften entlang der y- und z-Achse. Somit führt eine Verschiebung des eindimensionalen Problemgebiets in y- oder z-Richtung zu keiner Änderung der Problemstellung.

Diese Eigenschaft des Gebiets wird genutzt um die dreidimensionale Wellengleichung auf eine eindimensionale Beschreibung überzuführen. Für diesen Fall sind die Permittivität  $\varepsilon(\mathbf{r})$  und die elektrische Leitfähigkeit  $\sigma(\mathbf{r})$  Funktionen, die nur noch von der Ausbreitungsrichtung x abhängig sind.

Räumlich wird das Problem in Ausbreitungsrichtung im Intervall  $[a, b]$  und im Zeitbereich  $[0, t_0]$  betrachtet. Die elektromagnetische Welle wird am linken Rand  $x=a$  harmonisch durch die Funktion  $w(t) \sim \cos(\omega t + \varphi)$ <sup>1</sup> angeregt. Die Randbedingung muss dabei zusätzlich so modifiziert werden, dass der Rand für den Anteil der nach links hinauslaufenden Welle transparent wirkt.

Für den rechten Rand  $x=b$  wird so wie bei der linken Randbedingung eine transparente Randbedingung verwendet, sodass die nach rechts laufende Welle, am Rand aus dem Problemgebiet hinausläuft.

---

<sup>1</sup>In der FEM-Berechnung wird am linken Rand ein phasenverschobener Sinus vorgegeben. Für die analytische Berechnung wird jedoch, mit der äquivalent phasenverschobenen Kosinusfunktion gearbeitet.

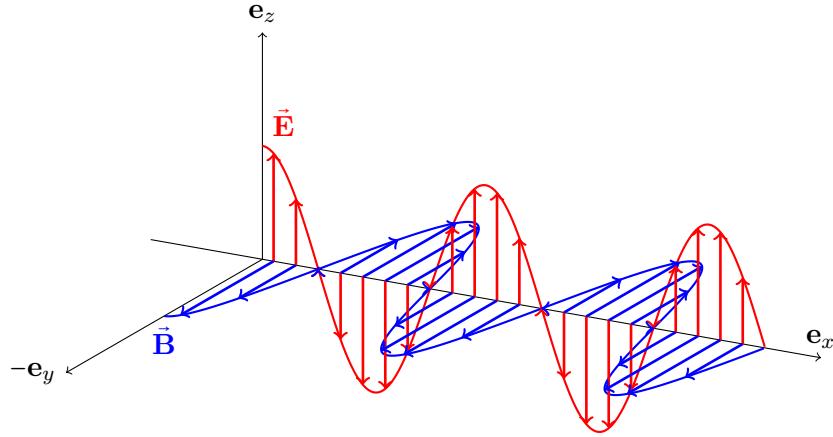


Abbildung 2.1: Eine sich in Vakuum ( $\epsilon_0, \mu_0$ ) ausbreitende ebene elektromagnetische Welle, welche sich mit Lichtgeschwindigkeit  $c_0$  in die positive x-Richtung bewegt.

Die eben beschriebene Problemstellung lässt sich durch die Gleichungen (2.1)-(2.5) mathematisch formulieren. Es ist zu erwähnen, dass  $u(x,t)$  die einkomponentige elektrische Feldstärke darstellt.

## 2.1 Wellengleichung

$$\frac{\partial^2}{\partial x^2}u(x,t) - \mu_0\epsilon(x)\frac{\partial^2}{\partial t^2}u(x,t) - \mu_0\sigma(x)\frac{\partial}{\partial t}u(x,t) = 0, \quad \forall (x,t) \in (a,b) \times (0,t_0] \quad (2.1)$$

## 2.2 Randbedingungen

$$\frac{\partial}{\partial x}u(a,t) - \sqrt{\mu_0\epsilon(a)}\frac{\partial}{\partial t}u(a,t) = -2\sqrt{\mu_0\epsilon(a)}\frac{\partial}{\partial t}w(t), \quad \forall t \in (0,t_0] \quad (2.2)$$

$$\frac{\partial}{\partial x}u(b,t) + \sqrt{\mu_0\epsilon(b)}\frac{\partial}{\partial t}u(b,t) = 0, \quad \forall t \in (0,t_0] \quad (2.3)$$

## 2.3 Anfangsbedingungen

$$\begin{aligned} u(x,0) &= 0, & \forall x \in (a,b] \\ u(a,0) &= w(0) \end{aligned} \quad (2.4)$$

$$\begin{aligned} \frac{\partial}{\partial t}u(x,0) &= 0, & \forall x \in (a,b] \\ \frac{\partial}{\partial t}u(a,0) &= \frac{\partial}{\partial t}w(0) \end{aligned} \quad (2.5)$$

# 3

## KAPITEL

# Herleitung der dreidimensionalen Wellengleichung

Ausgehend von den Maxwell-Gleichungen in einem quellenfreien Raumteil ( $\rho = 0, \mathbf{J}_0 = \mathbf{0}$ )

$$\nabla \times \mathbf{E} = -\frac{\partial}{\partial t} \mathbf{B} \quad (3.1)$$

$$\nabla \cdot \mathbf{D} = 0 \quad (3.2)$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial}{\partial t} \mathbf{D} \quad (3.3)$$

$$\nabla \cdot \mathbf{B} = 0 \quad (3.4)$$

mit den Materialbeziehungen für ein lineares, inhomogenes und isotropes Medium

$$\mathbf{D} = \epsilon(\mathbf{r})\mathbf{E} \quad (3.5)$$

$$\mathbf{B} = \mu(\mathbf{r})\mathbf{H} \quad (3.6)$$

und dem lokalen ohmschen Gesetz

$$\mathbf{J} = \sigma(\mathbf{r})\mathbf{E} \quad (3.7)$$

wird im Folgenden die dreidimensionale Wellengleichung der elektrischen Feldstärke hergeleitet. Durch diese Gleichung ist das Verhalten der elektromagnetischen Welle im Medium bestimmt.

Nimmt man die Rotation des Induktionsgesetzes (3.1)<sup>1</sup>, setzt den Ampere-Maxwell-Satz (3.3) ein und berücksichtigt anschließend das lokale ohmsche Gesetz (3.7), so erhält man:

$$\nabla \times (\nabla \times \mathbf{E}) = \nabla \times \left[ -\frac{\partial}{\partial t} \mathbf{B} \right] \quad (3.8)$$

$$= -\frac{\partial}{\partial t} \left[ \nabla \times \mu(\mathbf{r}) \mathbf{H} \right] \quad (3.9)$$

$$= -\frac{\partial}{\partial t} \left[ (\nabla \mu(\mathbf{r})) \times \mathbf{H} + \mu(\mathbf{r}) \underbrace{\nabla \times \mathbf{H}}_{\sigma(\mathbf{r}) \mathbf{E} + \frac{\partial}{\partial t} \mathbf{D}} \right] \quad (3.10)$$

Nach kurzem Umschreiben ergibt sich:

$$\nabla \times (\nabla \times \mathbf{E}) = -\mu(\mathbf{r})\sigma(\mathbf{r}) \frac{\partial}{\partial t} \mathbf{E} - \mu(\mathbf{r})\varepsilon(\mathbf{r}) \frac{\partial^2}{\partial t^2} \mathbf{E} + \underbrace{\left( \frac{\partial}{\partial t} \mathbf{B} \right)}_{-\nabla \times \mathbf{E}} \times \frac{\nabla \mu(\mathbf{r})}{\mu(\mathbf{r})} \quad (3.11)$$

Verwendet man nun für die linke Seite die Identität  $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \mathbf{b}(\mathbf{a} \cdot \mathbf{c}) - \mathbf{c}(\mathbf{a} \cdot \mathbf{b})$  und berücksichtigt, dass beide Differentialoperatoren auf  $\mathbf{E}$  wirken, so folgt daraus

$$\nabla(\nabla \cdot \mathbf{E}) - \nabla^2 \mathbf{E} = -\mu(\mathbf{r})\sigma(\mathbf{r}) \frac{\partial}{\partial t} \mathbf{E} - \mu(\mathbf{r})\varepsilon(\mathbf{r}) \frac{\partial^2}{\partial t^2} \mathbf{E} - (\nabla \times \mathbf{E}) \times \frac{\nabla \mu(\mathbf{r})}{\mu(\mathbf{r})} \quad (3.12)$$

$$\Leftrightarrow \nabla^2 \mathbf{E} - \nabla(\nabla \cdot \mathbf{E}) = \mu(\mathbf{r})\sigma(\mathbf{r}) \frac{\partial}{\partial t} \mathbf{E} + \mu(\mathbf{r})\varepsilon(\mathbf{r}) \frac{\partial^2}{\partial t^2} \mathbf{E} + (\nabla \times \mathbf{E}) \times \frac{\nabla \mu(\mathbf{r})}{\mu(\mathbf{r})} \quad (3.13)$$

Der Ausdruck  $\nabla(\nabla \cdot \mathbf{E})$  aus (3.13) wird mithilfe der Materialbeziehung aus Gleichung (3.5) zu  $\nabla(\nabla \cdot \frac{\mathbf{D}}{\varepsilon(\mathbf{r})})$  umgeschrieben und nach Anwendung der Produktregel mit dem Satz vom elektrischen Hullenfluss (3.2) für ( $\rho=0$ ) vereinfacht. Somit erhält man  $\nabla(\nabla(\frac{1}{\varepsilon(\mathbf{r})}) \cdot \mathbf{D})$ . Nach Verwendung der Verknüpfungsbeziehung (3.5) und Anwendung der Quotientenregel kommt man schlussendlich auf  $-\nabla(\frac{\nabla(\varepsilon(\mathbf{r}))}{\varepsilon(\mathbf{r})} \cdot \mathbf{E})$ .

Eingesetzt und umgeformt ergibt sich die dreidimensionale Wellengleichung welche die Ausbreitung des elektrischen Feldes in einem inhomogenen Medium mit der Permittivität  $\varepsilon(\mathbf{r})$  und Permeabilität  $\mu(\mathbf{r})$  beschreibt. (vgl.[HMRH13], Glg. (7)).

$$\nabla^2 \mathbf{E} - \mu(\mathbf{r})\varepsilon(\mathbf{r}) \frac{\partial^2}{\partial t^2} \mathbf{E} = \mu(\mathbf{r})\sigma(\mathbf{r}) \frac{\partial}{\partial t} \mathbf{E} + (\nabla \times \mathbf{E}) \times \frac{\nabla \mu(\mathbf{r})}{\mu(\mathbf{r})} - \nabla \left( \frac{\nabla(\varepsilon(\mathbf{r}))}{\varepsilon(\mathbf{r})} \cdot \mathbf{E} \right) \quad (3.14)$$

Durch die Beziehung  $\frac{\nabla f(\mathbf{r})}{f(\mathbf{r})} = \nabla(\ln(f(\mathbf{r}))$  können die Terme, in denen eine einfache Ableitung der Permittivität oder Permeabilität auftritt, kompakter angeschrieben werden.

---

<sup>1</sup>Nach dem Satz von Schwarz kann für die stetig differenzierbare magnetische Feldstärke  $\mathbf{B}$  die Reihenfolge der zeitlichen und räumlichen Differentiation vertauscht werden.

# KAPITEL 4

## Eindimensionale Wellengleichung

Es handelt sich bei dem Problem um ein in y- und z-Richtung verschiebungsinvariantes Gebiet. Dieser Vorteil wird genutzt um die dreidimensionale Wellengleichung auf eine eindimensionale Gleichung zu reduzieren. Die Permittivität  $\varepsilon(\mathbf{r})$  und die elektrische Leitfähigkeit  $\sigma(\mathbf{r})$  sind damit nur noch von der Ortsvariable x abhängig. Mit der Konsequenz, dass sich bei senkrechter Polarisation des elektrischen Feldes ( $\mathbf{e}_z$  – Richtung) der Ausdruck  $\frac{\nabla(\varepsilon(\mathbf{r}))}{\varepsilon(\mathbf{r})} \cdot \mathbf{E}$  der Wellengleichung (3.14) zu Null ergibt. Zusätzlich wird die Permeabilität mit  $\mu(\mathbf{r}) = \mu_0$  angenommen, da dies auf die meisten realen Anwendungen zutrifft.

Mit dieser Annahme ist die Ableitung  $\frac{\nabla\mu(\mathbf{r})}{\mu(\mathbf{r})}$  des Terms  $(\nabla \times \mathbf{E}) \times \frac{\nabla\mu(\mathbf{r})}{\mu(\mathbf{r})}$  ebenfalls Null. Es verschwindet der gesamte Term aus der Wellengleichung (3.14).

Durch Berücksichtigung der eben besprochenen Annahmen vereinfacht sich die dreidimensionale Wellengleichung aus (3.14) zu

$$\nabla^2 \mathbf{E} - \mu_0 \varepsilon(x) \frac{\partial^2}{\partial t^2} \mathbf{E} - \mu_0 \sigma(x) \frac{\partial}{\partial t} \mathbf{E} = 0 \quad (4.1)$$

Die elektrische Feldstärke  $\mathbf{E}$  besitzt lediglich eine Komponente in z-Richtung und lässt sich daher als  $\mathbf{E} = u(x, t) \mathbf{e}_z$  anschreiben.

Es folgt die eindimensionale Wellengleichung, welche für die numerische Berechnung mit der Finite-Elemente-Methode (FEM) herangezogen wird.

$$\frac{\partial^2}{\partial x^2} u(x, t) - \mu_0 \varepsilon(x) \frac{\partial^2}{\partial t^2} u(x, t) - \mu_0 \sigma(x) \frac{\partial}{\partial t} u(x, t) = 0 \quad (4.2)$$

Aus dem Induktionsgesetz (3.1) und der Tatsache, dass die elektrische Feldstärke  $\mathbf{E}$  ausschließlich eine z-Komponente besitzt, folgt ein rein in  $\mathbf{e}_y$  gerichtetes magnetisches Feld  $\mathbf{B}$ .



# KAPITEL 5

## Randbedingungen

Nachdem im Kapitel 3 und 4 die Wellengleichung beschrieben wurde, welche nur das Verhalten der Welle im Medium beschreibt, müssen nun um ein konkretes Problem zu erhalten, Anfangs- und Randbedingungen vorgegeben werden. Wie schon in (2.1)-(2.5) zu finden ist, sind die beiden Randbedingungen des eindimensionalen Problemgebiets so gewählt, dass der auslaufende Anteil der Welle am Rand  $x=a$  und  $x=b$  nicht reflektiert wird.

### 5.1 Rechter Rand - nichtreflektierend

Es wird bei der Beschreibung der beiden Randbedingungen zuerst der rechte Rand behandelt, da das Ergebnis für die linke Randbedingung übernommen werden kann und diese eine etwas aufwendigere Behandlung benötigt.

Die Aufgabe ist eine Randbedingung für den rechten Rand  $x=b$  zu finden, sodass der nach rechts laufende Teil der Welle bei  $x=b$  nicht reflektiert wird und aus dem Problemgebiet ungehindert hinauslaufen kann. Hierfür wird ausgehend von der eindimensionalen Wellengleichung aus (4.2) der d'Alembert-Operator für eine ins Problemgebiet ein- und auslaufende Welle faktorisiert (vgl. [Arn07], S.12). Der erste Faktor behandelt dabei die nach rechts laufende (linker Rand) und der zweite die nach links laufende (rechter Rand) Welle, welche für die Randbedingung auf Null gesetzt wird.<sup>1</sup>

---

<sup>1</sup>Die Pfeile unter der geschwungenen Klammer signalisieren die linke bzw. rechte Randbedingung.

Aus der Faktorisierung des d'Alembert-Operators der Wellengleichung aus (4.2) folgt:

$$\left[ \frac{\partial^2}{\partial x^2} - \mu_0 \varepsilon(x) \frac{\partial^2}{\partial t^2} \right] u(x, t) - \mu_0 \sigma(x) \frac{\partial}{\partial t} u(x, t) = 0 \quad (5.1)$$

$$\underbrace{\left( \frac{\partial}{\partial x} - \sqrt{\mu_0 \varepsilon(x)} \frac{\partial}{\partial t} \right)}_{\leftarrow} \underbrace{\left( \frac{\partial}{\partial x} + \sqrt{\mu_0 \varepsilon(x)} \frac{\partial}{\partial t} \right)}_{\rightarrow} u(x, t) - \mu_0 \sigma(x) \frac{\partial}{\partial t} u(x, t) = 0 \quad (5.2)$$

(5.3)

Damit die nach rechts auslaufende Welle am rechten Rand bei  $x=b$  nicht reflektiert wird, muss der zweite Faktor der obigen Wellengleichung auf  $u(x, t)$  angewandt (nach links, zurück ins Problemgebiet laufender Anteil) Null sein, daraus ergibt sich:

$$\frac{\partial}{\partial x} u(b, t) + \sqrt{\mu_0 \varepsilon(b)} \frac{\partial}{\partial t} u(b, t) = 0, \quad \forall t \in (0, t_0] \quad (5.4)$$

## 5.2 Linker Rand - nichtreflektierend

Bei der Randbedingung für den linken Rand muss einerseits beachtet werden, dass der Anteil der im Problemgebiet gestreuten Welle, welcher sich nach links laufend ausbreitet, am linken Rand  $x=a$  nichtreflektiert aus dem Gebiet hinausläuft. Dieser Teil der Welle wird im Weiteren als Streubetrag  $u_s$  bezeichnet (engl. scattering). Das Feld am linken Rand ergibt sich aus der Superposition der gestreuten Welle mit der harmonischen Anregung  $w(t)$ .

$$u(a, t) = u_s(a, t) + w(t) \quad (5.5)$$

$$\frac{\partial}{\partial x} u_s(a, t) - \sqrt{\mu_0 \varepsilon(a)} \frac{\partial}{\partial t} u_s(a, t) = 0, \quad \forall t \in (0, t_0] \quad (5.6)$$

Weiters gilt zu berücksichtigen, dass sich die am linken Rand angeregte Welle vollständig nach rechts in das Problemgebiet ausbreitet, sodass sie eine von links einlaufende homogene ebene Welle darstellt:

$$\frac{\partial}{\partial x} w(t) + \sqrt{\mu_0 \varepsilon(a)} \frac{\partial}{\partial t} w(t) = 0 \quad (5.7)$$

Addiert man nun die beiden Gleichungen aus (5.6) und (5.7), ergänzt die Summe mit  $\pm\sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}w(t)$ , so ergibt sich:

$$\frac{\partial}{\partial x}u(a,t) - \sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}u_s(a,t) + \sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}w(t) \pm \sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}w(t) = 0 \quad (5.8)$$

$$\frac{\partial}{\partial x}u(a,t) - \sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}u(a,t) + 2\sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}w(t) = 0 \quad (5.9)$$

Somit ist mit

$$\frac{\partial}{\partial x}u(a,t) - \sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}u(a,t) = -2\sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}w(t), \quad \forall t \in (0, t_0] \quad (5.10)$$

die Randbedingung für den linken Rand beschrieben.



# 6

## KAPITEL

# Finite-Elemente-Methode - lineare Basisfunktionen

Da das Anfangsrandwertproblem, wie es in (2.1)-(2.5) formuliert ist, sowohl partielle Ableitung nach dem Ort als auch nach der Zeit besitzt, wird das Problem in zwei Schritten gelöst. Zuerst wird es durch die FEM im Ort diskretisiert und anschließend das erhaltene Gleichungssystem mit einem Zeitschrittverfahren gelöst.

Den Ausgangspunkt für die FEM bietet die schwache Formulierung, die die Anforderung an die Differenzierbarkeit der Lösung herabsetzt. Um das Problem nun räumlich zu diskretisieren wird es in  $n$  äquidistante Teilintervalle unterteilt, sogenannte finite Elemente (FE), welche jeweils einen Anfangs- und Endknoten besitzen (Intervallanfang bzw. Intervallende). Zu erwähnen ist, dass die Unterteilung in FE im Allgemeinen nicht äquidistant sein muss. Im Weiteren werden  $(n+1)$  Basisfunktionen  $p_i(x)$  für  $i = 0, \dots, n$  eingeführt und auf jeweils einen Knoten  $i$  der finiten Elemente definiert. Diese Basisfunktionen besitzen wichtige Eigenschaften, die im Folgenden noch behandelt werden.

Die Näherungslösung  $u_h(x, t)$  und die Testfunktion  $\varphi_h(x)$  werden als Linearkombination der Basisfunktionen  $p_i(x)$  dargestellt. Die Aufgabe der Finiten-Elemente-Methode besteht darin ein Gleichungssystem zu erhalten mit dem diese Koeffizienten bestmöglich bestimmt werden. Dafür setzt man die Näherungslösung  $u_h(x, t)$  in die schwache Formulierung ein und erhält daraus ein Gleichungssystem bestehend aus schwach besetzten Matrizen. Dieses wird unter Berücksichtigung der gegebenen Anfangswerte (Anfangszustand (2.4) und Anfangsgeschwindigkeit (2.5)) mithilfe eines Zeitschrittverfahrens gelöst.

## 6.1 Die schwache Formulierung

Der Vorteil der schwachen Formulierung liegt darin, dass die Anforderung an die Differenzierbarkeit der Lösung herabgesetzt wird. Dabei wird die Differentialgleichung aus

(2.1) mit einer Testfunktion  $\varphi(x)$  multipliziert und anschließend über das gesamte Problemgebiet integriert. Durch die Partielle Integration wird eine einfache Ableitung im Ort der Lösung  $u(x, t)$  an die Testfunktion  $\varphi(x)$  abgegeben. Die Testfunktion  $\varphi(x)$  ist eine Funktion, welche einen kompakten Träger besitzt und beliebig oft stetig differenzierbar ist.

Dabei ist  $\varphi$  aus dem Raum  $V_0 = \{\varphi \in H^1[a, b]\}$ .

Die Wellengleichung

$$\frac{\partial^2}{\partial x^2} u(x, t) - \mu_0 \varepsilon(x) \frac{\partial^2}{\partial t^2} u(x, t) - \mu_0 \sigma(x) \frac{\partial}{\partial t} u(x, t) = 0 \quad (6.1)$$

mit der Testfunktion  $\varphi(x)$  multipliziert und anschließend über das gesamte Problemgebiet integriert, ergibt

$$\int_a^b \left[ u''(x, t) \varphi(x) - \mu_0 \varepsilon(x) \frac{\partial^2}{\partial t^2} u(x, t) \varphi(x) - \mu_0 \sigma(x) \frac{\partial}{\partial t} u(x, t) \varphi(x) \right] dx = 0. \quad (6.2)$$

Im Weiteren wird aus Gründen der Übersichtlichkeit für die räumliche Ableitung  $\frac{\partial}{\partial x}$  ein Hochkomma ' geschrieben.

Führt man für den ersten Term des Integrals die partielle Integration

$$\int_a^b [u''(x, t) \varphi(x)] dx = \underbrace{(u'(x, t) \varphi(x))|_a^b}_{u'(b, t) \varphi(b) - u'(a, t) \varphi(a)} - \int_a^b [u'(x) \varphi'(x)] dx \quad (6.3)$$

durch, so verliert man sozusagen eine einfache Ableitung der Lösung  $u(x, t)$  an die Testfunktion  $\varphi(x, t)$ .

Bei dem Auswerten des ersten Terms (6.3) nach der Partiellen Integration werden die Randbedingungen des linken und rechten Randes aus (5.10) und (5.4) in die Gleichung (6.3) eingebracht.

Es ergibt sich für das Integral (6.3):

$$\begin{aligned} \int_a^b [u''(x, t) \varphi(x)] dx &= -\sqrt{\mu_0 \varepsilon(b)} \frac{\partial}{\partial t} u(b, t) \varphi(b) - \sqrt{\mu_0 \varepsilon(a)} \frac{\partial}{\partial t} u(a, t) \varphi(a) + \\ &\quad 2\sqrt{\mu_0 \varepsilon(a)} \frac{\partial}{\partial t} w(t) \varphi(a) - \int_a^b [u'(x, t) \varphi'(x)] dx \end{aligned} \quad (6.4)$$

Nutzt man diese Zusammenhänge und setzt sie in Gleichung (6.2) ein, so erhält man:

$$\begin{aligned}
 & \int_a^b [u'(x,t)\varphi'(x)]dx + \int_a^b [\mu_0\varepsilon(x)\frac{\partial^2}{\partial t^2}u(x,t)\varphi(x)]dx + \\
 & \int_a^b [\mu_0\sigma(x)\frac{\partial}{\partial t}u(x,t)\varphi(x)]dx + \left[ \sqrt{\mu_0\varepsilon(b)}\frac{\partial}{\partial t}u(b,t)\varphi(b) \right] + \\
 & \left[ \sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}u(a,t)\varphi(a) \right] = \left[ 2\sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}w(t)\varphi(a) \right]
 \end{aligned} \tag{6.5}$$

**Es folgt die schwache Formulierung des Problems:**

Gesucht ist

$$u(x,t) \in V = \{u(x,t) \in H^1[a,b], t \in (0,t_0]\},$$

sodass

$$a(u, \varphi) = F(\varphi), \forall \varphi \in V_0 = \{\varphi \in H^1[a,b]\}$$

gilt, wobei

$$\begin{aligned}
 a(u, \varphi) &:= \int_a^b [u'(x,t)\varphi'(x)]dx + \frac{\partial^2}{\partial t^2} \int_a^b [\mu_0\varepsilon(x)u(x,t)\varphi(x)]dx + \\
 &\quad \frac{\partial}{\partial t} \int_a^b [\mu_0\sigma(x)u(x,t)\varphi(x)]dx + \frac{\partial}{\partial t} \left[ \sqrt{\mu_0\varepsilon(b)}u(b,t)\varphi(b) \right] + \\
 &\quad \frac{\partial}{\partial t} \left[ \sqrt{\mu_0\varepsilon(a)}u(a,t)\varphi(a) \right] \\
 F(\varphi) &:= \left[ 2\sqrt{\mu_0\varepsilon(a)}\frac{\partial}{\partial t}w(t)\varphi(a) \right]
 \end{aligned} \tag{6.6}$$

(vgl. [JL13],[Hol20]).

## 6.2 Finite Elemente - Räumliche Diskretisierung

Das eindimensionale Problemgebiet in x-Richtung  $x \in [a,b]$  wird in n äquidistante Teilintervalle unterteilt, sogenannte finite Elemente.

Man erhält nach der Unterteilung des Intervalls in n finite Elemente eine Schrittweite von:

$$h_x = \frac{b-a}{n} \tag{6.7}$$

Jedes der  $n$  finiten Elemente besitzt jeweils einen Anfangs- und Endknoten, die von  $i = 0, \dots, n$  durchnummiert sind:

$$x_{i+1} = x_i + h_x \quad (6.8)$$

Intervalle der  $n$  finiten Elemente des Problemgebiets  $\Omega$ :

$$\begin{aligned} T_i &= [x_{i-1}, x_i] \\ \Omega &= \bigcup_{i=1}^n T_i \end{aligned} \quad (6.9)$$

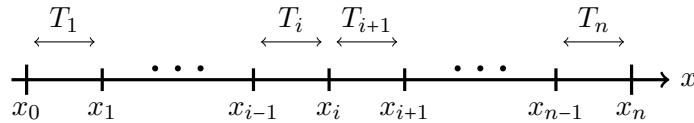


Abbildung 6.1: Unterteilung des Problemgebiets  $[a,b]$  in  $n$  finite Elemente.

### 6.3 Näherungslösung für lineare Basisfunktionen

Die Näherungslösung  $u_h(x, t)$  und die Testfunktion  $\varphi_h(x)$  werden als Linearkombination von linearen Basisfunktionen (Hutfunktionen)  $p_i(x)$  für  $i = 0, \dots, n$  ausgedrückt. Überlagert man nach dem Lösen des Problems, also der Bestimmung der Koeffizienten  $u_i$  die mit  $u_i$  gewichteten Basisfunktionen, so erhält man als Näherungslösung einen Polygonzug mit den Knoten  $x_i$  als Stützstellen.<sup>1</sup>

Jedem Knoten  $i$  der  $(n+1)$  Knoten des Problemgebiets ist eine Basisfunktion  $p_i(x)$  zugeordnet. Die Basisfunktionen sind in (6.12)-(6.14) definiert. (vgl. [Hol20]).

$$n \in \mathbb{N}$$

$$u_h(x, t) = \sum_{j=0}^n (u_j(t) p_j(x)) \quad (6.10)$$

$$\varphi_h(x) = \sum_{i=0}^n (\varphi_i p_i(x)) \quad (6.11)$$

---

<sup>1</sup>Im Kapitel 7 findet man die Erweiterung auf eine Berechnung mit Basisfunktionen höherer Ordnung.

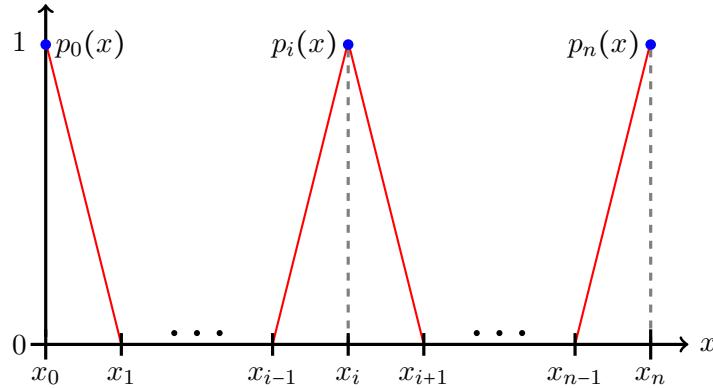


Abbildung 6.2: Lineare Basisfunktionen  $p_0(x)$  bis  $p_n(x)$  (Hutfunktionen).

Die Basisfunktionen  $p_i(x)$  für  $i = 0, \dots, n$  sind definiert als:

$$p_0(x) = \begin{cases} \frac{x_1-x}{h}, & x_0 \leq x \leq x_1 \\ 0, & x_1 < x \leq x_n \end{cases} \quad (6.12)$$

$$p_j(x) = \begin{cases} 0, & x_0 \leq x \leq x_{j-1} \\ \frac{x-x_{j-1}}{h}, & x_{j-1} < x \leq x_j \\ \frac{x_{j+1}-x}{h}, & x_j < x \leq x_{j+1} \\ 0, & x_{j+1} < x \leq x_n \end{cases} \quad (6.13)$$

$$p_n(x) = \begin{cases} 0, & x_0 \leq x \leq x_{n-1} \\ \frac{x-x_{n-1}}{h}, & x_{n-1} < x \leq x_n \end{cases} \quad (6.14)$$

Da in der schwachen Formulierung (6.6) auch die einfach abgeleiteten Basisfunktionen auftreten, müssen diese auch definiert werden.

$$p'_0(x) = \begin{cases} -\frac{1}{h}, & x_0 \leq x \leq x_1 \\ 0, & x_1 < x \leq x_n \end{cases} \quad (6.15)$$

$$p'_j(x) = \begin{cases} 0, & x_0 \leq x \leq x_{j-1} \\ \frac{1}{h}, & x_{j-1} < x \leq x_j \\ -\frac{1}{h}, & x_j < x \leq x_{j+1} \\ 0, & x_{j+1} < x \leq x_n \end{cases} \quad (6.16)$$

$$p'_n(x) = \begin{cases} 0, & x_0 \leq x \leq x_{n-1} \\ \frac{1}{h}, & x_{n-1} < x \leq x_n \end{cases} \quad (6.17)$$

Die Basisfunktionen  $p_i(x)$  mit  $i = 0, \dots, n$  besitzen folgende wichtige Eigenschaften:

- Sie sind nur in dem Knoten auf den sie definiert sind Eins.  $\Leftrightarrow p_i(x_i) = 1$ .
- Sie verschwinden in jedem anderen Knoten.  $\Leftrightarrow p_i(x_j) = 0, j \neq i$ .
- Sie sind auf maximal zwei benachbarten FE ungleich Null.

Setzt man die Näherungslösung  $u_h(x, t)$  und die Testfunktionen  $\varphi_h(x)$  in die Gleichung der schwache Formulierung aus (6.6) ein, so erhält man:

$$\begin{aligned} a(u_h, \varphi_h) &:= \int_a^b [u'_h(x, t)\varphi'_h(x)] dx + \frac{\partial^2}{\partial t^2} \int_a^b [\mu_0 \varepsilon(x) u_h(x, t) \varphi_h(x)] dx + \\ &\quad \frac{\partial}{\partial t} \int_a^b [\mu_0 \sigma(x) u_h(x, t) \varphi_h(x)] dx + \frac{\partial}{\partial t} [\sqrt{\mu_0 \varepsilon(b)} u_h(b, t) \varphi_h(b)] + \\ &\quad \frac{\partial}{\partial t} [\sqrt{\mu_0 \varepsilon(a)} u_h(a, t) \varphi_h(a)] \\ F(\varphi) &:= \left[ 2\sqrt{\mu_0 \varepsilon(a)} \frac{\partial}{\partial t} w(t) \varphi_h(a) \right] \end{aligned} \quad (6.18)$$

Dies kann aufgrund der Bilinearität von  $a(u_h, \varphi_h)$  in beiden Argumenten  $u_h(x, t)$  und  $\varphi_h(x)$  und der Linearität von  $F(\varphi)$  in  $\varphi_h(x)$  zu

$$\begin{aligned} i &= 0, \dots, n \\ \sum_{j=0}^n & \left[ \int_a^b [p'_i(x) p'_j(x)] dx + \frac{\partial^2}{\partial t^2} \int_a^b [\mu_0 \varepsilon(x) p_i(x) p_j(x)] dx + \right. \\ & \left. \frac{\partial}{\partial t} \int_a^b [\mu_0 \sigma(x) p_i(x) p_j(x)] dx + \frac{\partial}{\partial t} [\sqrt{\mu_0 \varepsilon(b)} \delta_{i,j} \delta_{i,n}] + \frac{\partial}{\partial t} [\sqrt{\mu_0 \varepsilon(a)} \delta_{i,j} \delta_{i,0}] \right] u_j(t) = \\ & \left[ 2\sqrt{\mu_0 \varepsilon(a)} \delta_{i,0} \frac{\partial}{\partial t} w(t) \right] \end{aligned} \quad (6.19)$$

umgeschrieben werden.

Fasst man nun alle  $(n+1)$  Gleichungen für  $i = 0, \dots, n$  zusammen, ergibt sich daraus das gewöhnliche Differentialgleichungssystem

$$\mathbf{K}\mathbf{u}_h(t) + \mathbf{M} \frac{d^2}{dt^2} \mathbf{u}_h(t) + \mathbf{C} \frac{d}{dt} \mathbf{u}_h(t) = \mathbf{f}(t) \quad (6.20)$$

mit den Matrizen  $\mathbf{K}$ ,  $\mathbf{M}$  und  $\mathbf{C} \in \mathbb{R}^{(n+1) \times (n+1)}$  und dem Vektor  $\mathbf{f} \in \mathbb{R}^{n+1}$

- **K**... Steifigkeitsmatrix
- **M**... Massenmatrix
- **C**... Dämpfungsmatrix
- **f** ... Lastvektor

wobei der Lösungsvektor  $\mathbf{u}_h(t)$  definiert ist als:

$$\mathbf{u}_h(t) = (u_0(t), u_1(t), \dots, u_{n-1}(t), u_n(t))^T \quad (6.21)$$

In (6.22)-(6.26) sind die Einträge der Matrizen und des Lastvektors angegeben. Es ist darauf hinzuweisen, dass die Indizierung bei Null beginnt.

$$[K_{i,j}]_{i,j=0}^n = \left[ \int_a^b [p'_i(x)p'_j(x)] dx \right]_{i,j=0}^n \quad (6.22)$$

$$[M_{i,j}]_{i,j=0}^n = \left[ \int_a^b [\mu_0 \varepsilon(x)p_i(x)p_j(x)] dx \right]_{i,j=0}^n \quad (6.23)$$

$$[C_{i,j}]_{i,j=0}^n = \left[ \int_a^b [\mu_0 \sigma(x)p_i(x)p_j(x)] dx + \right. \quad (6.24)$$

$$\left. \sqrt{\mu_0 \varepsilon(a)} \delta_{i,j} \delta_{i,0} + \sqrt{\mu_0 \varepsilon(b)} \delta_{i,j} \delta_{i,n} \right]_{i,j=0}^n \quad (6.25)$$

$$[f_i(t)]_{i=0}^n = \left[ 2\sqrt{\mu_0 \varepsilon(a)} \frac{\partial}{\partial t} w(t) \delta_{i,0} \right]_{i=0}^n \quad (6.26)$$

## 6.4 Numerische Integration

Nach der schwachen Formulierung des Problems und Einsetzen der Näherungslösung, erhält man ein Gleichungssystem, welches durch Matrix-Vektor-Schreibweise in (6.20) dargestellt ist. Es entsteht dadurch die Aufgabe die Einträge der Matrizen **K**, **M** und **C** und des Vektors  $\mathbf{f}(t)$  zu berechnen. Diese sind in den Berechnungsvorschriften (6.22)-(6.26) zu finden. Da man in der Vorschrift zur Berechnung der Matrizen **M** (6.23) und **C** (6.25) aufgrund der Materialeigenschaften  $\varepsilon(x)$  und  $\sigma(x)$  im Allgemeinen einen beliebig ortsabhängigen Integranden vorfindet, werden die Matrixeinträge numerisch berechnet. In diesem Kapitel wird nur auf die numerische Integration bei linearen Basisfunktionen (Hutfunktionen) eingegangen. Die Ergänzung auf Basisfunktionen höherer Ordnung ist in Kapitel 7 zu finden.

Für die numerische Quadratur bietet sich die Gauß-Quadratur an, da sie durch geeignete Wahl der Stützstellen und Gewichte einen relativ hohen Konvergenzgrad besitzt. Man kann mit der Gauß-Quadratur N-ten Grades ein Polynom vom Grad  $(2N-1)$  exakt integrieren.

#### 6.4.1 Quadraturformel

Kurz zusammengefasst wird bei der numerischen Integration einer Funktion  $f(x)$  über dem Intervall  $[a, b]$  die Funktion bei vorgegebenen Stützstellen  $x_i$  ausgewertet und mit einem Gewicht  $w_i$ , welches zur Stützstelle gehört multipliziert. Summiert man über all diese gewichteten Funktionswerte, so erhält man eine Näherungslösung des Integrals. Dabei ist zu erwähnen, dass die Summe der Gewichte der Intervalllänge entsprechen muss. Dies muss offensichtlich gelten, da ansonsten eine konstante Funktion  $f(x) = c \in \mathbb{R}$  nicht exakt integriert wird.

Das eben beschriebene Verfahren lässt sich mit folgender Quadraturformel

$$Q(f) = \sum_{i=1}^N w_i f(x_i) \quad (6.27)$$

berechnen. Der Näherungswert für die Integration einer beliebigen Funktion  $f(x)$  über dem Referenzelement  $[-1, 1]$  erhält man durch:

$$\int_{-1}^1 f(x) dx \approx \sum_{i=1}^N w_i f(x_i) \quad (6.28)$$

Die Quadraturregel dafür, mit ihren Stützstellen  $x_i$  und Gewichten  $w_i$  ist in (6.4.3) zu finden.

#### 6.4.2 Berechnung der Matrix- und Vektoreinträge

Bevor die Gauß-Legendre-Quadraturregel genauer beschrieben wird, wird zunächst näher auf die eigentliche Berechnung eingegangen.

Um die Einträge der Matrizen und des Vektors aus (6.22)-(6.26) zu berechnen muss wie angegeben das jeweilige Integral gelöst werden. (vgl. [Hol20]).

Aus den Eigenschaften der linearen Basisfunktionen  $p_i(x)$ , wie sie nach Definition (6.12)-(6.14) beschrieben sind folgt, dass nur Einträge der Haupt- und der ersten Nebendiagonale der Matrizen  $\mathbf{K}$ ,  $\mathbf{M}$  und  $\mathbf{C}$  ungleich Null sein können. Da sozusagen bei der Berechnung der Einträge, die Zeilen und Spalten der Matrix die Basisfunktionen selektieren, welche miteinander multipliziert werden. Als Beispiel ist für den Eintrag  $(i, j)$  die Basisfunktion  $p_i(x)$  mit der Basisfunktion  $p_j(x)$  zu multiplizieren.

Das Produkt aus zwei Basisfunktionen  $p_i(x)p_j(x)$  für  $i, j = 0, \dots, n$  ist nur ein von Null verschiedener Wert, wenn es sich um die gleichen oder um benachbarte Basisfunktionen

handelt, d.h.  $|i - j| \leq 1$ . Es folgt eine Matrix mit Tridiagonalstruktur.<sup>2</sup>

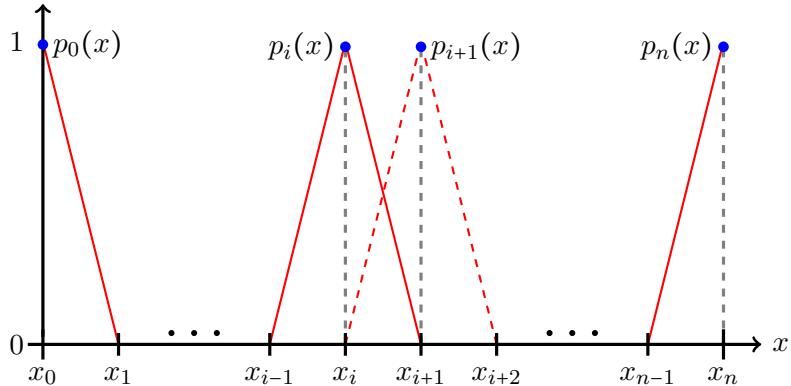


Abbildung 6.3: Globale Basisfunktionen  $p_i(x)$ .

Es ergeben sich wie in Matrix  $\mathbf{A}$  (6.29) dargestellt nur die markierten Einträge (gleicher oder benachbarter Basisfunktionen) der drei Matrizen  $\mathbf{K}$ ,  $\mathbf{M}$  und  $\mathbf{C}$  von Null verschieden. Die Asterisk-Schriftzeichen auf der Hauptdiagonale stehen dabei für die Einträge bei denen die Basisfunktionen  $p_i(x)$  bzw. deren Ableitungen  $p'_i(x)$  quadriert werden. Asterisk-Schriftzeichen auf den ersten Nebendiagonalen entsprechen dem Produkt aus zwei benachbarten Basisfunktionen  $p_i(x)p_j(x)$  bzw. dem Produkt aus deren Ableitungen  $p'_i(x)p'_j(x)$  für  $i \neq j$  mit  $|i - j| = 1$ .

Globale Matrix  $\mathbf{A}$ :

$$\mathbf{A} = \begin{pmatrix} * & * & & & & 0 \\ * & * & * & & & \\ * & * & & & & \\ & & \ddots & & & \\ & & & * & * & \\ & & & * & * & * \\ 0 & & & * & * & \end{pmatrix} \quad (6.29)$$

Ziel ist es, die Einträge der globalen Matrix  $\mathbf{A}$  aus (6.29) durch eine möglichst übersichtliche Berechnung zu erhalten. Dafür wird die globale Matrix aus (6.29) in  $n$   $2 \times 2$  Submatrix-Blöcke unterteilt, welche schon in (6.29) gut zu sehen sind. Der  $i$ -te  $2 \times 2$  Subblock wird Elementmatrix  $\mathbf{A}_i$  für das  $i$ -te finite Element genannt, da seine vier Einträge aus Integralen über dem  $i$ -ten finiten Element bestimmt werden. Zusätzlich ist darauf

---

<sup>2</sup>Tridiagonalstruktur ergibt sich nur bei linearen Basisfunktionen (Hutfunktionen). Vorteil der Tridiagonalstruktur ist, dass effizientere Algorithmen zum Lösen des Gleichungssystems existieren. So ein Algorithmus ist in Kapitel 8.2 beschrieben.

hinzuweisen, dass sich diese lokalen Elementmatrizen global gesehen bei ihren Hauptdiagonalelementen überlappen. Dies muss beim Eintragen der lokalen Elementmatrizen  $\mathbf{A}_i$  in die globale Matrix  $\mathbf{A}$  berücksichtigt werden.

Weiters wird das Integral, welches für die Berechnung der Einträge der lokalen Elementmatrix  $\mathbf{A}_i$  zu lösen ist, nicht über das i-te finite Element  $[x_i, x_{i+1}]$ , sondern auf dem Referenzelement  $[-1,1]$  berechnet.

Dafür wird das Integral über  $[x_i, x_{i+1}]$  mithilfe einer Koordinatentransformation auf ein Integral über  $[-1, 1]$  transformiert. Dieser Zusammenhang ist für den allgemeinen Fall mithilfe des Transformationssatzes (6.30) beschrieben.

$$\int_{\Phi(\Omega)} f(y) dy = \int_{\Omega} f(\Phi(x)) \det(\mathbf{J}(\Phi(x))) dx \quad (6.30)$$

Die Transformation eines Integrals über dem i-ten finiten Element  $[x_i, x_{i+1}]$  auf das Referenzelement  $[-1,1]$  ergibt sich daher zu

$$\int_{x_i}^{x_{i+1}} f(x) dx = \int_{-1}^{+1} f(x(\xi)) \left| \frac{dx(\xi)}{d\xi} \right| d\xi \quad (6.31)$$

mit der Transformation zwischen den Intervallen

$$x(\xi) = ih_x + h_x \frac{(\xi + 1)}{2} \quad (6.32)$$

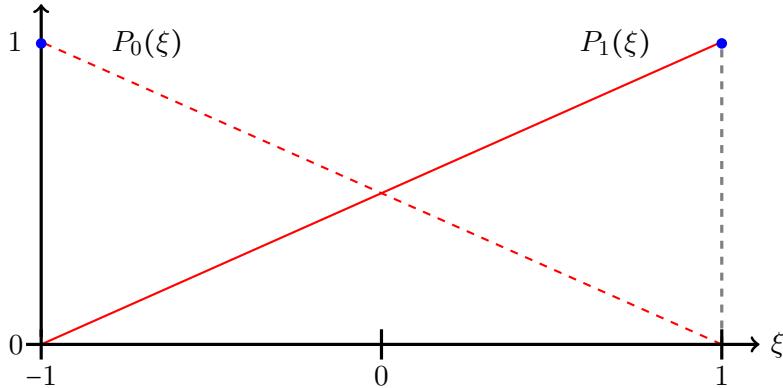
wobei  $h_x = \frac{b-a}{n}$  die FE-Länge ist.

Für die Integration über dem Referenzelement werden zwei lokale Basisfunktionen  $P_0(\xi)$  und  $P_1(\xi)$  auf  $[-1,1]$  definiert, welche in Abb. 6.4 zu sehen sind.

Für die Matrix  $\mathbf{K}$  muss weiters darauf geachtet werden, dass bei Transformation der abgeleiteten Basisfunktionen die innere Ableitung berücksichtigt wird.

$$\begin{aligned} j &= 0, 1, \dots, n-1, n; k = 0 \text{ oder } 1 \\ \frac{dp_j(x(\xi))}{dx} &= \frac{dp_j(x(\xi))}{d\xi} \frac{d\xi(x)}{dx} = \frac{dP_k(\xi)}{d\xi} \frac{2}{h_x} \end{aligned} \quad (6.33)$$

Nach Berechnung der n lokalen Elementmatrizen  $\mathbf{A}_i$  müssen diese in die globale Matrix  $\mathbf{A}$  eingetragen werden. Dies geschieht über Tabelle 6.5, die den Zusammenhang der Einträge der Elementmatrizen  $\mathbf{A}_i$  und der globalen Matrix  $\mathbf{A}$  für  $i = 0, \dots, n-1$  darstellt.


 Abbildung 6.4: Lokale Basisfunktionen  $P_0(\xi)$  und  $P_1(\xi)$ .

lokaler Index	globaler Index
0	i
1	i+1

Abbildung 6.5: Zusammenhang der Einträge der lokalen und globalen Matrix.

Das gerade eben besprochene Verfahren zur Berechnung der Matrixeinträge der globalen Matrix wird genutzt um die Einträge der Matrizen  $\mathbf{K}$ ,  $\mathbf{M}$  und  $\mathbf{C}$  zu bestimmen. Der Vorteil der systematischen Berechnung der Einträge zeigt sich insbesondere bei Verwendung von Basisfunktionen höherer Ordnung und ist [Hol20] und [Whi17] beschrieben.

### 6.4.3 Gauß-Legendre-Quadratur

Die Stützstellen  $x_i$  gemeinsam mit den Gewichten  $w_i$  für  $i = 1, \dots, N$  bilden die Quadraturregel nach Gauß, welche Polynome vom Grad kleiner gleich  $(2N-1)$  perfekt integriert. Diese hohe Konvergenzordnung erhält man indem man die Stützstellen und Gewichte passend wählt. Im Folgenden wird auf die Berechnung der Quadraturregel näher eingegangen. Die  $N$  Stützstellen  $x_i$  der Gauß-Legendre-Quadratur berechnen sich für  $i = 1, \dots, N$  als Nullstellen des Legendre-Polynoms  $N$ -ten Grades. Weiters wird gezeigt wie sich die Nullstellen des Legendre-Polynoms  $N$ -ten Grades über ein Eigenwertproblem lösen lassen. Dieser Ansatz ist im Golub-Welsch-Algorithmus [GW69] zu finden.

Legendre-Polynome sind orthogonale Polynome, definiert auf dem Intervall  $[-1, 1]$  mit einer Gewichtsfunktion, die 1 entspricht. Daraus folgt die Eigenschaft, dass alle Nullstellen im Intervall  $(-1, 1)$  liegen.

Betrachtet man nun die Polynombasis  $\{l_i(x)|_{i=0}^N\}$  der ersten  $(N+1)$  Legendre-Polynome, so kann jedes Legendre-Polynom  $l_i(x)$  mit  $i \geq 1$  mithilfe der Dreier-Term-Rekursion aus den zwei vorherigen Legendre-Polynomen  $l_{i-1}(x)$  und  $l_{i-2}(x)$  berechnet werden.

$$i = 1, \dots, N; l_{-1} = 0; l_0 = 1 \\ l_i(x) = \frac{2i-1}{i} xl_{i-1}(x) - \frac{i-1}{i} l_{i-2}(x), i \geq 1 \quad (6.34)$$

Mit dieser Rekursionsvorschrift berechnen sich die ersten Legendre-Polynome für  $l_0 = 1$  zu:

- $l_1 = x$
- $l_2 = \frac{1}{2}(3x^2 - 1)$
- $l_3 = \frac{1}{2}(5x^3 - 3x)$

Aus der Rekursionsvorschrift der ersten  $(N+1)$  Legendre-Polynome  $\{l_i(x)|_{i=0}^N\}$  erhält man somit  $N$  Gleichungen, die in Matrix-Vektor-Schreibweise zu dem Gleichungssystem aus (6.35) zusammenfasst werden.<sup>3</sup>

$$x \begin{bmatrix} l_0(x) \\ l_1(x) \\ \vdots \\ l_{N-2}(x) \\ l_{N-1}(x) \end{bmatrix} = \begin{bmatrix} 0 & b_1 & & & \\ a_2 & 0 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{N-1} & 0 & b_{N-1} \\ & & & a_N & 0 \end{bmatrix} \begin{bmatrix} l_0(x) \\ l_1(x) \\ \vdots \\ l_{N-2}(x) \\ l_{N-1}(x) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ l_N(x) \end{bmatrix} \quad (6.35)$$

Zu einer Tridiagonalmatrix  $\mathbf{T}$  und  $\mathbf{e}_N = (0, 0, \dots, 0, 1)^T$  zusammengefasst, ergibt sich (6.35) zu:

$$x\mathbf{l}(x) = \mathbf{T}\mathbf{l}(x) + l_N(x)\mathbf{e}_N \quad (6.36)$$

Betrachtet man nun das Gleichungssystem aus (6.35), so bemerkt man, dass das Auffinden der  $N$  Nullstellen von  $l_N(x)$  ein Eigenwertproblem darstellt. Es muss damit  $l_N(x) = 0$  wird, ein Eigenwert  $x_i$  multipliziert mit dem Vektor  $\mathbf{l}$  das Matrix-Vektor-Produkt  $\mathbf{T}\mathbf{l}$  ergeben. Dabei sind  $x_i$  die  $N$  Eigenwerte der Matrix und auch die Nullstellen des Legendre-Polynoms  $N$ -ten Grades. Es zeigt sich also, dass sich die Nullstellensuche des Legendre-Polynoms  $N$ -ten Grades auf ein Eigenwertproblem überführen lässt.

Um die Eigenwerte der Matrix  $\mathbf{T}$  zu bestimmen wäre es effizienter den QR-Algorithmus zu verwenden (vgl. [GW69]). Darauf wurde jedoch verzichtet und stattdessen der bereits

---

<sup>3</sup>Die Koeffizienten  $a_i$  und  $b_i$  ergeben sich aus der Drei-Term-Rekursion (6.34).

implementierte Algorithmus aus einer linearen Algebra Bibliothek verwendet.

Diese Vereinfachung ist vertretbar da der Grad der Gauß-Quadratur nicht allzu hoch ist und die Quadraturformel maximal dreimal berechnet werden muss. Man kann für die Berechnung der Matrizen  $\mathbf{K}$ ,  $\mathbf{M}$  und  $\mathbf{C}$  jeweils einen unterschiedlichen Grad der Quadratur wählen. Dies ist sinnvoll, wenn kompliziertere Funktionen für  $\varepsilon(x)$  oder  $\sigma(x)$  vorgegeben werden. Für die Steifigkeitsmatrix  $\mathbf{K}$  ist ein Quadratgrad von eins ausreichend, da nur maximal über eine Konstante (Produkt aus abgeleiteten Basisfunktionen  $p'_i(x)$ ) integriert wird. Zusätzlich ist zu erwähnen, dass die Stützstellen und Gewichte der Gauß-Quadratur symmetrisch auftreten. Somit muss man sie nur auf einer der beiden Seiten berechnen, entweder  $(-1,0]$  oder  $[0, 1)$ .

Die zu den Nullstellen gehörenden Gewichte lassen sich über die Formel

$$w_i = \frac{2}{(1 - x_i^2)[l'_N(x_i)]^2} \quad (6.37)$$

berechnen, welche in ([AS72], S. 887) zu finden ist.

Um das Gewicht  $w_i$  aus der Formel (6.37) zu bestimmen, muss die Ableitung des N-ten Legendre-Polynoms ausgewertet bei der Stützstelle  $x_i$  berechnet werden. Die Ableitung des Legendre-Polynoms N-ten Grades  $l'_N(x)$  wird berechnet, indem über die Drei-Term-Rekursion die Legendre-Polynome bis zum N-ten Grad als Spaltenvektor in einer Matrix gespeichert werden, anschließend bei dem letzten Spaltenvektor eine Schiebeoperation und Multiplikation durchgeführt wird, welche der einmaligen Ableitung entspricht.

Der Python-Quellcode um die Gauß-Legendre-Quadraturregel N-ten Grades zu berechnen ist in Kapitel 10.4 zu finden.

In ([AS72], S. 916-919, Tab. 25.4) sind die Stützstellen und Gewichte der Gauß-Quadratur tabelliert. Für die zweite und dritte Ordnung der Gauß-Legendre Quatraturformel stimmt die Berechnung durch den Quellcode aus 10.4 auf alle 15 angegebenen Nachkommastellen mit der Tabelle überein.

Die Tabelle 6.1 beschreibt die Auswertung der Quadraturformel für N=2 und N=3.

$\pm x_i$	$w_i$	$\pm x_i$	$w_i$
0.577350269189626	1	0.0000000000000000	0.8888888888888889
		0.774596669241483	0.5555555555555556

Tabelle 6.1: Gauß-Legendre Quadraturregel für (a) N=2 und (b) N=3.

Wie bereits erwähnt erreicht man mit der Gauß-Quadratur für besonders glatte Funktionen, auch schon bei niedriger Ordnung, recht gute Ergebnisse.

In Abb. 6.6 ist jedoch eine für die numerische Integration sehr ungünstige Funktion gewählt. Abb. 6.7 zeigt für steigende Ordnung der Quadraturformel die Konvergenz des Näherungswertes.

Zusätzlich erkennt man bei dem Konvergenzverhalten aus Abb. 6.7, dass eine Stützstelle bei der Unstetigkeitsstelle  $x=0$  einen genaueren Wert liefert. Dies tritt bei ungerader Anzahl der Stützstellen auf und behandelt den Fall, dass der Intervallübergang genau bei der Unstetigkeitsstelle liegt.

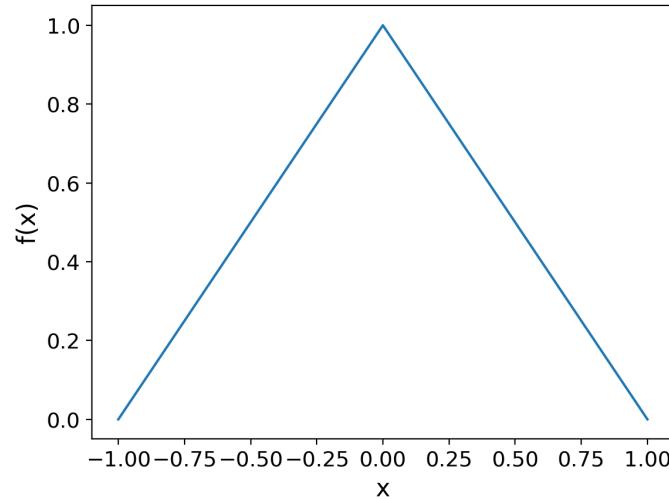


Abbildung 6.6: Hutfunktion über dem Referenzelement  $[-1, 1]$ .

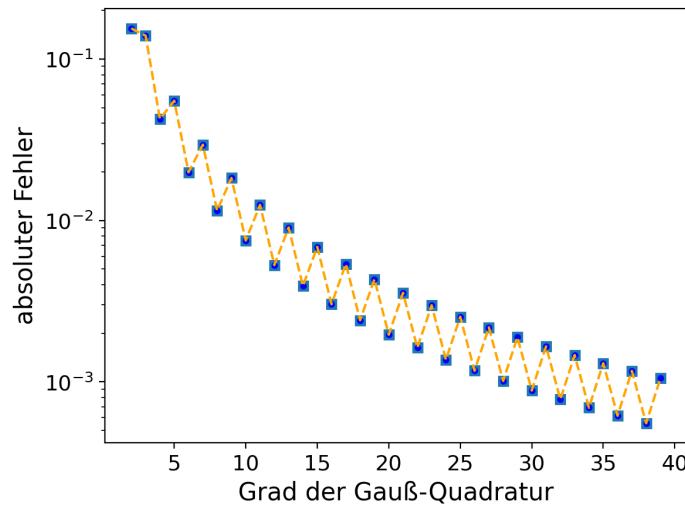


Abbildung 6.7: Konvergenz der Gaußquadratur für die Funktion aus Abb. 6.6.

# Höhere Ordnungen der Basisfunktionen

Es ist nicht nur wie in Kapitel 6 beschrieben möglich die Näherungslösung  $u_h(x, t)$  als Linearkombination von linearen Basisfunktionen  $p_i(x)$  darzustellen, sondern dafür auch Basisfunktionen  $p_i(x)$  höherer Ordnung zu verwenden. Es ergibt sich dabei über jedem der n FE eine stückweise Interpolation höherer Ordnung.

Hierfür werden Stützstellen innerhalb jedes finiten Elements hinzugefügt und Basisfunktionen darauf definiert. Diese weiteren Basisfunktionen höherer Ordnung sind nur auf dem jeweiligen finiten Element ungleich Null und liefern deshalb nur in der Elementmatrix des jeweiligen finiten Elements Einträge ungleich Null.

Für die Basisfunktionen werden im Weiteren Lagrange- bzw. Legendre-Polynome höherer Ordnung genutzt, welche in (7.1) bzw. (7.6) definiert sind.

Dadurch, dass die Berechnung bis auf ein paar Details gleich wie die Berechnung mit linearen Basisfunktionen (Hutfunktionen) aufgebaut ist, wird in diesem Kapitel das bereits Besprochene nur um das Notwendigste ergänzt.

Als erstes stellt sich die Frage wie viele innere Knoten für jedes FE zusätzlich auftreten. Bei der Verwendung von Basisfunktionen N-ter Ordnung treten in jedem FE (N-1) zusätzliche innere Knoten auf. Somit ergeben sich aus den inneren und äußeren Knoten jedes finiten Elements (N+1) Knoten. Mit den darauf definierten Basisfunktionen bekommt man durch Überlagern das Interpolationspolynom N-ter Ordnung des jeweiligen finiten Elements.

Als Beispiel dafür ist in Abb. 7.1 das (k+1)-te FE mit quadratischen Lagrange-Polynomen dargestellt.

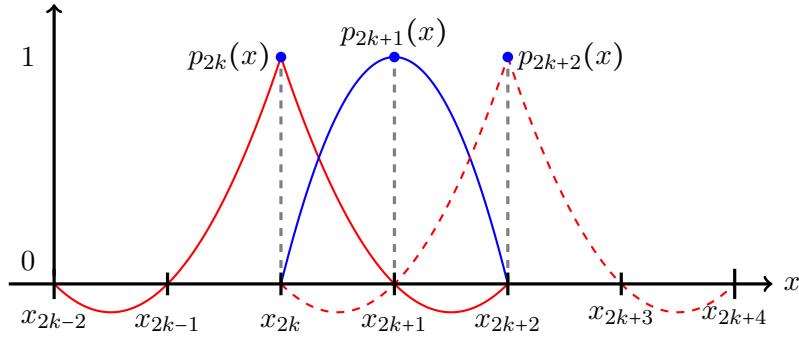


Abbildung 7.1: Das  $(k+1)$ -te finite Element mit den zugehörigen quadratischen Lagrange Basisfunktionen.

## 7.1 Lagrange Polynombasis

In Gleichung (7.1) ist die Berechnungsvorschrift der Lagrange-Polynome  $N$ -ter Ordnung für  $i = 0, \dots, N$  definiert.

$$L_{N,i}(\xi) = \prod_{j=0, j \neq i}^N \frac{\xi - \xi_j}{\xi_i - \xi_j} \quad (7.1)$$

Die abgeleiteten Lagrange-Polynome  $N$ -ter Ordnung ergeben sich zu:

$$l_{N,i}(\xi) = \sum_{m=0, m \neq i}^N \left[ \frac{1}{\xi_i - \xi_m} \prod_{j=0, j \neq (i,m)}^N \frac{\xi - \xi_j}{\xi_i - \xi_j} \right] \quad (7.2)$$

In Abb. 7.2 bzw. Abb. 7.3 sind die aus (7.1) und (7.2) definierten Lagrange-Polynome bzw. abgeleiteten Lagrange-Polynome vierter Ordnung dargestellt.

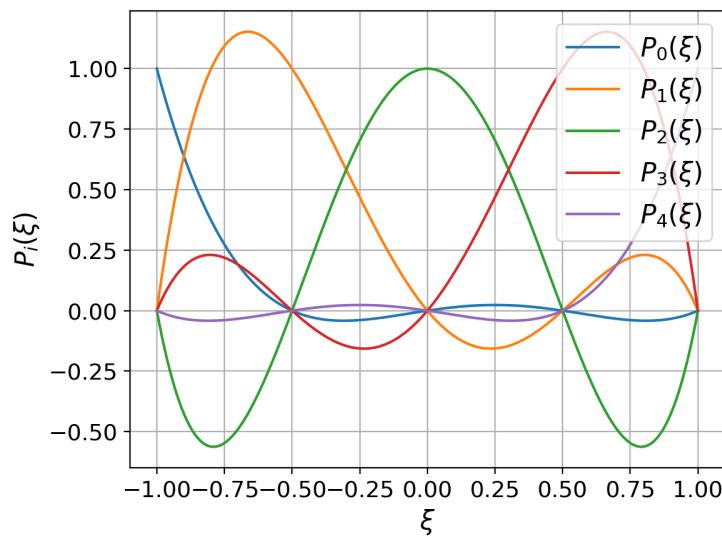


Abbildung 7.2: Lagrange-Polynome vierter Ordnung.

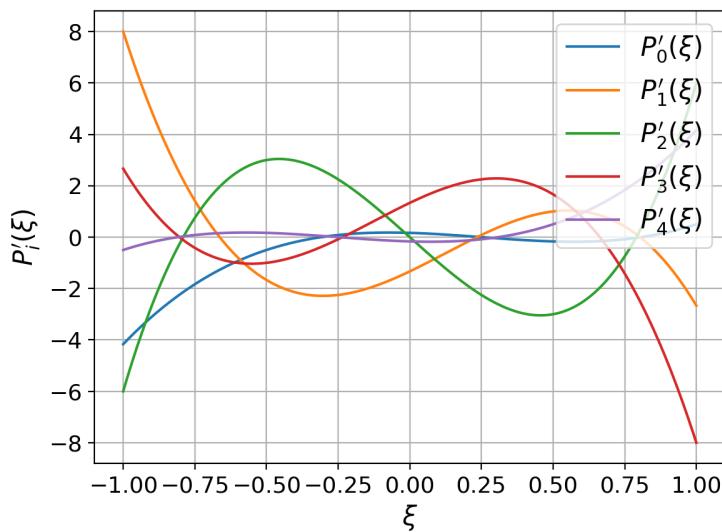


Abbildung 7.3: Abgeleitete Lagrange-Polynome vierter Ordnung.

Der Python-Quellcode für die Berechnung der Lagrange-Polynome und der abgeleiteten Lagrange-Polynome N-ter Ordnung ist in 10.6 zu finden.

## 7.2 Hierarchische Polynombasis

Wie schon in (6.34) definiert lautet die Drei-Term-Rekursion für die ersten (N+1) Legendre-Polynome (vgl. [Hol20], S.112-115):

$$\begin{aligned} i &= 1, \dots, N; l_{-1} = 0; l_0 = 1 \\ l_i(\xi) &= \frac{2i-1}{i} \xi l_{i-1}(\xi) - \frac{i-1}{i} l_{i-2}(\xi), i \geq 1 \end{aligned} \quad (7.3)$$

Die integrierten Legendre-Polynome lassen sich als

$$L_i(\xi) \int_{-1}^{\xi} l_{i-1}(\xi') d\xi', i \geq 1 \quad (7.4)$$

anschreiben. Daraus folgt die 3-Term-Rekursion der integrierten Legendre-Polynome:

$$\begin{aligned} i &= 1, \dots, N; L_1 = \xi; L_2(\xi) = \frac{1}{2}(\xi^2 - 1) \\ (i+1)L_{i+1}(\xi) &= (2i-1)\xi L_i(\xi) - (i-2)L_{i-1}(\xi), i \geq 2 \end{aligned} \quad (7.5)$$

Aus den linearen Basisfunktionen und den integrierten Legendre-Polynomen erhält man die hierarchische Basis N-ter Ordnung. Dabei ist darauf zu achten, dass die Nummerierung der Basisfunktionen bei Null beginnt.

$$\begin{aligned} i &= 1, \dots, N-1; L_1 = \xi; L_2(\xi) = \frac{1}{2}(\xi^2 - 1) \\ L_{i+1}(\xi) &= \frac{(2i-1)\xi}{(i+1)} L_i(\xi) - \frac{(i-2)}{(i+1)} L_{i-1}(\xi), i \geq 2 \\ P_0 &= \frac{1-\xi}{2}; P_j(\xi) = L_{j+1}(\xi) \text{ für } 0 < j < N; P_N(\xi) = \frac{1+\xi}{2} \end{aligned} \quad (7.6)$$

Die abgeleiteten Basisfunktionen  $P'_j(\xi)$  sind für  $0 < j < N$  die Legendre-Polynome.

In Abb. 7.4 bzw. Abb. 7.5 sind die Basisfunktionen der hierarchischen Basis und die zugehörigen abgeleiteten Basisfunktionen dargestellt.

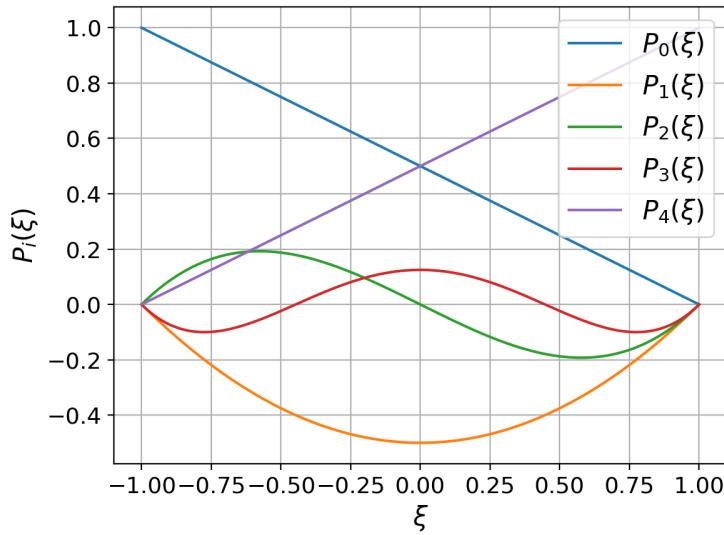


Abbildung 7.4: Basisfunktionen der hierarchischen Basis für  $N=5$ .

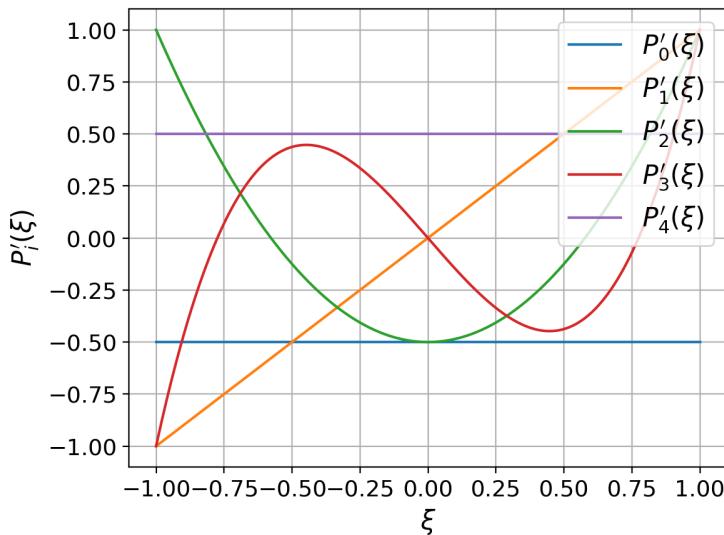


Abbildung 7.5: Abgeleitete Basisfunktionen der hierarchischen Basis für  $N=5$ .

Der Python-Quellcode für die Berechnung der Basisfunktionen und der abgeleiteten Basisfunktionen der hierarchischen Basis  $N$ -ter Ordnung ist in 10.6 zu finden.

### 7.3 Weitere Berechnung

Die zuvor definierten Basisfunktionen für eine Interpolation N-ter Ordnung über jedem FE werden im Weiteren auf die FE-Knoten definiert. Nach Ansetzen der Näherungslösung als Linearkombination der Basisfunktionen und Einsetzen in die schwache Formulierung des Problems, erhält man ein Differentialgleichungssystem wie es in (6.20) dargestellt ist.

Es muss wie im Kapitel 6 um die Einträge der Matrizen  $\mathbf{M}$  und  $\mathbf{C}$  zu bestimmen das Integral über dem Produkt von Basisfunktionen bzw. für die Matrix  $\mathbf{K}$  der abgeleiteten Basisfunktionen berechnet werden. Aus den Eigenschaften der Basisfunktionen folgt eine schwach besetzte globale Matrix  $\mathbf{A}$ , welche aus Submatrix-Blöcken (Elementmatrix  $\mathbf{A}_i$ ) der finiten Elemente besteht. Der Unterschied zu der globalen Matrix aus (6.29) ist, dass nun die Elementmatrizen durch die höhere Ordnung der Basisfunktionen höhere Dimensionen besitzen.

Da nur die Basisfunktionen der äußeren Knoten über maximal zwei FE ungleich Null sein können, überlappen die Elementmatrizen  $\mathbf{A}_i$  in der globalen Matrix nur dort wo die quadrierten Basisfunktionen der äußeren Knoten auftreten. Auf dieses muss beim Eintragen geachtet werden.

Für das Eintragen der lokalen Elementmatrizen  $\mathbf{A}_i$  in die globale Matrix  $\mathbf{A}$  für die finiten Elemente  $i = 0, \dots, n - 1$  wird die Tabelle 7.1 herangezogen.

lokaler Index	globaler Index
0	2i
1	2i+1
2	2i+2
:	:
N	2i+N

Tabelle 7.1: Zusammenhang der lokalen und globalen Matrix für Basisfunktionen N-ter Ordnung.

Nach Berechnen der drei Matrizen  $\mathbf{K}$ ,  $\mathbf{M}$  und  $\mathbf{C}$  und des Vektors  $\mathbf{f}(t)$  ergibt sich ein gewöhnliches Differentialgleichungssystem wie aus (6.20), welches nur mehr von der Zeit abhängt und in Kapitel 8 mit einem Zeitschrittverfahren gelöst wird.

Es ist darauf hinzuweisen, dass nach Erhalt der Lösung der Lösungsvektor

$$\mathbf{u}_h(t) = (u_0(t_k), u_1(t_k), \dots, u_{nN-1}(t_k), u_{nN}(t_k))^T, \quad k = 0, \dots, m \quad (7.7)$$

(nN+1) Koeffizienten enthält, mit denen die Basisfunktionen gewichtet werden. Für eine Auswertung der Näherungslösung muss dabei nur das jeweilige FE mit seinen (N+1) Koeffizienten betrachtet werden. Die Implementierung einer solchen Methode findet man in 10.7.

# KAPITEL

# 8

## Lösung des gew. Differentialgleichungssystems - zeitliche Diskretisierung

### 8.1 Newmark-beta-Verfahren

In Kapitel 6 wurde mithilfe der FEM das Differentialgleichungssystem aus (6.20) bestimmt. Gegenüber der Wellengleichung aus (2.1) ist das Differentialgleichungssystem nun im Ort diskretisiert und beinhaltet ausschließlich Ableitungen nach der Zeit. Um weiters die Näherungslösung  $u_h(x, t)$  zu erhalten, wird das gewöhnliche Differentialgleichungssystem

$$\mathbf{K}\mathbf{u}_h(t) + \mathbf{M} \frac{d^2}{dt^2} \mathbf{u}_h(t) + \mathbf{C} \frac{d}{dt} \mathbf{u}_h(t) = \mathbf{f}(t) \quad (8.1)$$

durch ein Zeitschrittverfahren gelöst.

Als Zeitschrittverfahren wird das Newmark-beta-Verfahren genutzt. Man gibt hier den Anfangswert  $\mathbf{u}(t=0)$  und die Anfangsgeschwindigkeit  $\frac{d}{dt} \mathbf{u}(t)|_{t=0}$  vor und berechnet sich für die darauffolgenden Zeitschritte die Näherungslösung des (i+1)-ten Zeitschrittes aus der Näherungslösung des i-ten Zeitschrittes. Dabei muss ein Gleichungssystem wie es im Folgenden beschrieben ist, gelöst werden. Eine genaue Herleitung und Beschreibung des Newmark-beta-Verfahrens findet man in [LP19].

Es muss um die Näherungslösung für den nächsten Zeitschritt (i+1) zu erhalten das Gleichungssystem

$$\mathbf{A}\mathbf{u}_{i+1} = \tilde{\mathbf{f}}_{i+1} \quad (8.2)$$

gelöst werden, wobei  $\Delta t$  die Dauer eines Zeitschrittes darstellt. Das Newmark-beta-Verfahren ist für die Parameter  $\gamma = 1/2$  und  $\beta = 1/4$  unbedingt stabil.

Die Matrix  $\mathbf{A}$  und der Vektor  $\tilde{\mathbf{f}}_{i+1}$  sind definiert als:

$$\mathbf{A} = \frac{\mathbf{M}}{\beta \Delta t^2} + \frac{\gamma \mathbf{C}}{\beta \Delta t} + \mathbf{K} \quad (8.3)$$

$$\begin{aligned} \tilde{\mathbf{f}}_{i+1} &= \mathbf{f}((i+1)\Delta t) + \mathbf{M} \left[ \frac{\mathbf{u}_i}{\beta \Delta t^2} + \frac{\mathbf{v}_i}{\beta \Delta t} + \left( \frac{1}{2\beta} - 1 \right) \mathbf{a}_i \right] + \\ &\quad \mathbf{C} \left[ \frac{\gamma \mathbf{u}_i}{\beta \Delta t} - \mathbf{v}_i \left( 1 - \frac{\gamma}{\beta} \right) - \Delta t \mathbf{a}_i \left( 1 - \frac{\gamma}{2\beta} \right) \right] \end{aligned} \quad (8.4)$$

Die Geschwindigkeit  $\mathbf{v}_i$  und die Beschleunigung  $\mathbf{a}_i$  lassen sich in folgender Form bestimmen:

$$\mathbf{a}_{i+1} = \frac{\mathbf{u}_{i+1} - \mathbf{u}_i}{\beta \Delta t^2} - \frac{\mathbf{v}_i}{\beta \Delta t} - \left( \frac{1}{2\beta} - 1 \right) \mathbf{a}_i \quad (8.5)$$

$$\mathbf{v}_{i+1} = \gamma \frac{\mathbf{u}_{i+1} - \mathbf{u}_i}{\beta \Delta t} + \mathbf{v}_i \left( 1 - \frac{\gamma}{\beta} \right) + \Delta t \mathbf{a}_i \left( 1 - \frac{\gamma}{2\beta} \right) \quad (8.6)$$

Das Newmark-beta-Verfahren in Python implementiert, ist in 10.5 zu finden.

## 8.2 Algorithmus zum Lösen eines Tridiagonalsystems

Dadurch, dass sich bei linearen Basisfunktionen Systemmatrizen  $\mathbf{K}$ ,  $\mathbf{C}$  und  $\mathbf{M}$  mit Tridiagonalstruktur ergeben, kann ein effizienterer Algorithmus zum Lösen des Gleichungssystems genutzt werden. Dieser lässt sich durch eine LU-Zerlegung beschreiben und ist in ([Dat10], S.162) zu finden.

Bei höheren Ordnungen der Basisfunktionen ist der Algorithmus nicht mehr anwendbar, da die Systemmatrizen in diesem Fall keine Tridiagonalstruktur aufweisen. Der Vorteil besteht darin, dass die Lösung eines Tridiagonalgleichungssystems mit  $\mathcal{O}(n)$ , statt wie bei der Gauß-Elimination mit  $\mathcal{O}(n^3)$ , berechnet wird.

Betrachtet man ein Tridiagonalgleichungssystem

$$\begin{pmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & \ddots & \ddots & c_{n-1} & \\ 0 & & a_n & b_n & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{pmatrix} \quad (8.7)$$

mit  $a_1 = 0$  und  $c_n = 0$ , berechnen sich die Koeffizienten  $c'_i$  und  $d'_i$  zu:

$$c'_i = \begin{cases} \frac{c_i}{b_i}, & i = 1 \\ \frac{c_i}{b_i - a_i c'_{i-1}}, & i = 2, \dots, n-1 \end{cases} \quad (8.8)$$

$$d'_i = \begin{cases} \frac{d_i}{b_i}, & i = 1 \\ \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}}, & i = 2, \dots, n \end{cases} \quad (8.9)$$

Den Lösungsvektor  $\mathbf{x}$  erhält man durch Rückwärtseinsetzen:

$$x_n = d'_n \quad (8.10)$$

$$x_i = d'_i - c'_i x_{i+1}, \quad i = (n-1), \dots, 1 \quad (8.11)$$

In 10.5 findet man für den Algorithmus den zugehörigen Python-Quellcode.



# 9

## KAPITEL

# Vergleich der Näherungslösung mit der analytischen Lösung

In den nächsten Unterkapiteln werden spezielle Ausbreitungssituationen behandelt, welche sich auch einfach analytisch berechnen lassen. Hierfür wird auf die Ausbreitung im Vakuum, die Reflexion an einem idealen Leiter und der Feldverlauf bei einem Sprung der Permittivität eingegangen. Dabei wird der Fehler der Näherungslösung zur analytischen Lösung sowohl bei unterschiedlicher Diskretisierung in Ort und Zeit als auch für höhere Ordnungen der Basisfunktionen untersucht. Das Konvergenzverhalten der Lagrange-Polynombasis gegenüber der hierarchischen Basis wird ebenfalls beobachtet. Sowie der räumliche und zeitliche Betrachtungsbereich als auch die Frequenz der Welle werden für die erste Simulation beschrieben, passend gewählt und für die nachfolgenden Simulationen ohne zusätzliche Erwähnung übernommen.

Folgende Simulationsparameter werden für das Lesen dieses Kapitels vorausgesetzt:

- n ... Anzahl der finiten Elemente.
- m ... Anzahl der Zeitschritte.
- „Lagrange“ bzw. „Legendre“ weist darauf hin, dass Lagrange- bzw. Legendre-Polynome für die FEM-Berechnung verwendet wurden.

## 9.1 Ausbreitung im Vakuum

Der einfachste Fall, welcher betrachtet werden kann, ist eine Ausbreitung im leeren Raum mit der elektrischen Feldkonstante  $\epsilon_0$  und der magnetischen Feldkonstante  $\mu_0$ . Beim Durchqueren des Gebiets kann die homogene ebene Welle mit nichts wechselwirken, sie

breitet sich ungehindert mit Lichtgeschwindigkeit  $c_0$  in positiver x-Richtung aus. Für diesen Fall ist in Abb. 2.1 der elektrische und magnetische Feldstärkeverlauf dargestellt.

Im Weiteren wird das Problem analytisch beschrieben und mit der Näherungslösung  $u_h(x, t)$  verglichen. Dieses einfache Problem wird genutzt um bei wenigen finiten Elementen auf die Approximation innerhalb der finiten Elemente einzugehen und dies grafisch zu veranschaulichen.

### 9.1.1 Analytische Beschreibung

Die von links einlaufende Welle wird durch den komplexen Amplitudenvektor

$$\tilde{\mathbf{E}}_i(x, \omega) = \tilde{\mathbf{E}}_{i,0} e^{-j k x} \quad (9.1)$$

beschrieben, für den  $k$  die Wellenzahl darstellt, welche die räumliche Periodizität beschreibt. Durch Realteilbildung erhält man den elektrischen Feldstärkeverlauf:<sup>1</sup>

$$\mathbf{E}_i(x, t) = \operatorname{Re}\{\tilde{\mathbf{E}}_{i,0} e^{-j k x} e^{j \omega t}\} \quad (9.2)$$

$$= \mathbf{E}_{i,0} \cos(\omega t - kx + \varphi) \quad (9.3)$$

### 9.1.2 Numerische Berechnung

Die Welle wird für diese Simulation mit einer Kreisfrequenz von  $\omega = 376.73 \text{ GHz}$  angeregt, sodass sich im Problemgebiet  $[0, 1 \text{ cm}]$  genau zwei Wellenlängen  $\lambda$  ausgeben. Für den gewählten Zeitbereich  $[0, 100 \text{ ps}]$  laufen in der gesamten Simulationszeit sechs Wellenlängen durch das Problemgebiet.

Aus der Dispersionsrelation und der Definition der Wellenlänge erhält man die Beziehung:

$$c_0 = \frac{\omega}{k} \quad (9.4)$$

$$k\lambda = 2\pi \quad (9.5)$$

Mit der Bedingung, dass genau 2 Wellenlängen  $\lambda$  im Problemgebiet liegen, ergibt sich die Kreisfrequenz zu

$$\omega = \frac{2\pi c_0}{\frac{b-a}{2}} = 376.73 \text{ GHz}. \quad (9.6)$$

---

<sup>1</sup>Mit der Tilde über der Größe wird auf eine komplexe Zahl hingewiesen. Die Amplitude ist hier  $\tilde{\mathbf{E}}_{i,0} = E_{i,0} e^{j\varphi}$ , da diese von der Anfangsphasenverschiebung abhängt.

Simulationsparameter:

- Problemgebiet  $[0, 1\text{ cm}]$
- Zeitbereich  $[0, 100\text{ ps}]$
- Kreisfrequenz  $\omega = 376.73\text{ GHz}$
- Anregung  $w(t) = \sin(\omega t + \varphi)$

Diese Parameter bleiben auch für die weiteren Simulationen unverändert.

Bei numerischen Berechnungen mit Kosinus- und Sinus-Anregung ist aufgefallen, dass Anfangswerte ungleich Null (Kosinus-Anregung) und Anfangsgeschwindigkeiten ungleich Null (Sinus-Anregung) zu einer schlechten Konvergenz des Zeitschrittverfahrens führt. Bei Anfangswerten ungleich Null tritt dabei eine Welligkeit zu Beginn auf, die auch für weitere Zeitschritte nicht gänzlich verschwindet. Bei einer Anfangsgeschwindigkeit ungleich Null hat man bei grober Diskretisierung und niedriger Ordnung der Basisfunktionen einen leichten Offset der Lösung, welcher für höhere Ordnungen der Basisfunktionen und höherer Anzahl der Finiten Elemente geringer ausfällt.

Aus diesem Grund wurde in der ersten Periode der Anregung eine quadratische Funktion gewählt, die bei dem Wert von  $1/\sqrt{2}$  in eine harmonische Anregung übergeht. Zusätzlich wird bei der Wahl der Funktion  $at^2$  darauf geachtet, dass die Ableitung der Funktion zum Zeitpunkt des Übergangs  $t_{phi}$  mit  $\frac{d}{dt}w(t = t_{phi}) = \omega/\sqrt{2}$  übereinstimmt. Die Simulationsergebnisse zeigen dabei das erwartete Konvergenzverhalten mit deutlich genaueren Ergebnissen.

Für die Berechnung der Parameter folgt:

$$at_{phi}^2 = \sin(\omega t_{phi} + \varphi) = \frac{1}{\sqrt{2}} \quad (9.7)$$

$$2at_{phi} = \omega \cos(\omega t_{phi} + \varphi) = \frac{\omega}{\sqrt{2}} \quad (9.8)$$

Dividiert man Gleichung (9.7) durch (9.8) so ergibt sich für den Zeitpunkt des Übergangs  $t_{phi}$ , den Koeffizienten  $a$  und die Phasenverschiebung  $\varphi$ :

$$t_{phi} = \frac{2}{\omega} \quad (9.9)$$

$$a = \frac{1}{\sqrt{2}t_{phi}^2} \quad (9.10)$$

$$\varphi = \frac{\pi}{4} - \omega t_{phi} \quad (9.11)$$

## 9. VERGLEICH DER NÄHERUNGSLÖSUNG MIT DER ANALYTISCHEN LÖSUNG

---

Die erste Periode der Anregung  $w(t)$  ist in Abb. 9.1 dargestellt.

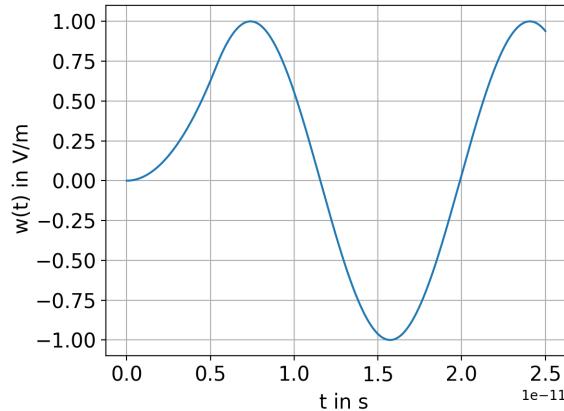


Abbildung 9.1: Anregung  $w(t)$  mit dem Anfangswert und der Anfangsgeschwindigkeit gleich Null.

In Abb. 9.2 ist der elektrische Feldstärkeverlauf für den Zeitpunkt  $\frac{t}{t_0} = 0.7$  zu sehen.

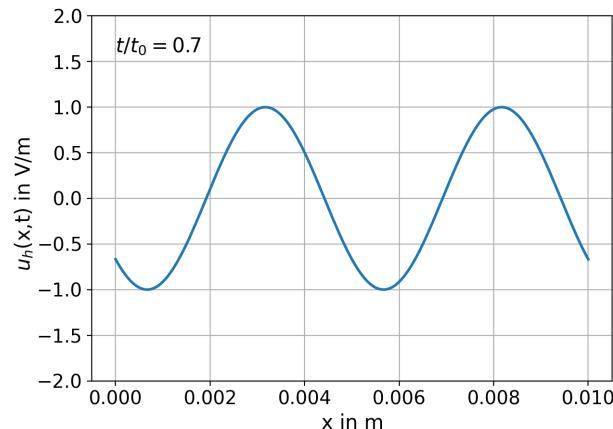


Abbildung 9.2: Lösung  $u_h(x,t)$  für den Zeitschritt  $\frac{m}{2}=500$  bei  $n=100$  mit Lagrange-Basisfunktionen dritter Ordnung.

In den Plots aus Abb. 9.3 sind die Lösungen für zwei finite Elemente bei unterschiedlicher Ordnung der Lagrange-Basisfunktionen für den Zeitschritt  $\frac{m}{2}$  dargestellt. Es soll bei dieser groben Diskretisierung im Ort (sprich ein finites Element pro Wellenlänge  $\lambda$ ) den Vorteil von Basisfunktionen höherer Ordnung darstellen. Dabei ist die Ordnung der Interpolation innerhalb der beiden finiten Elementen gut zu erkennen.

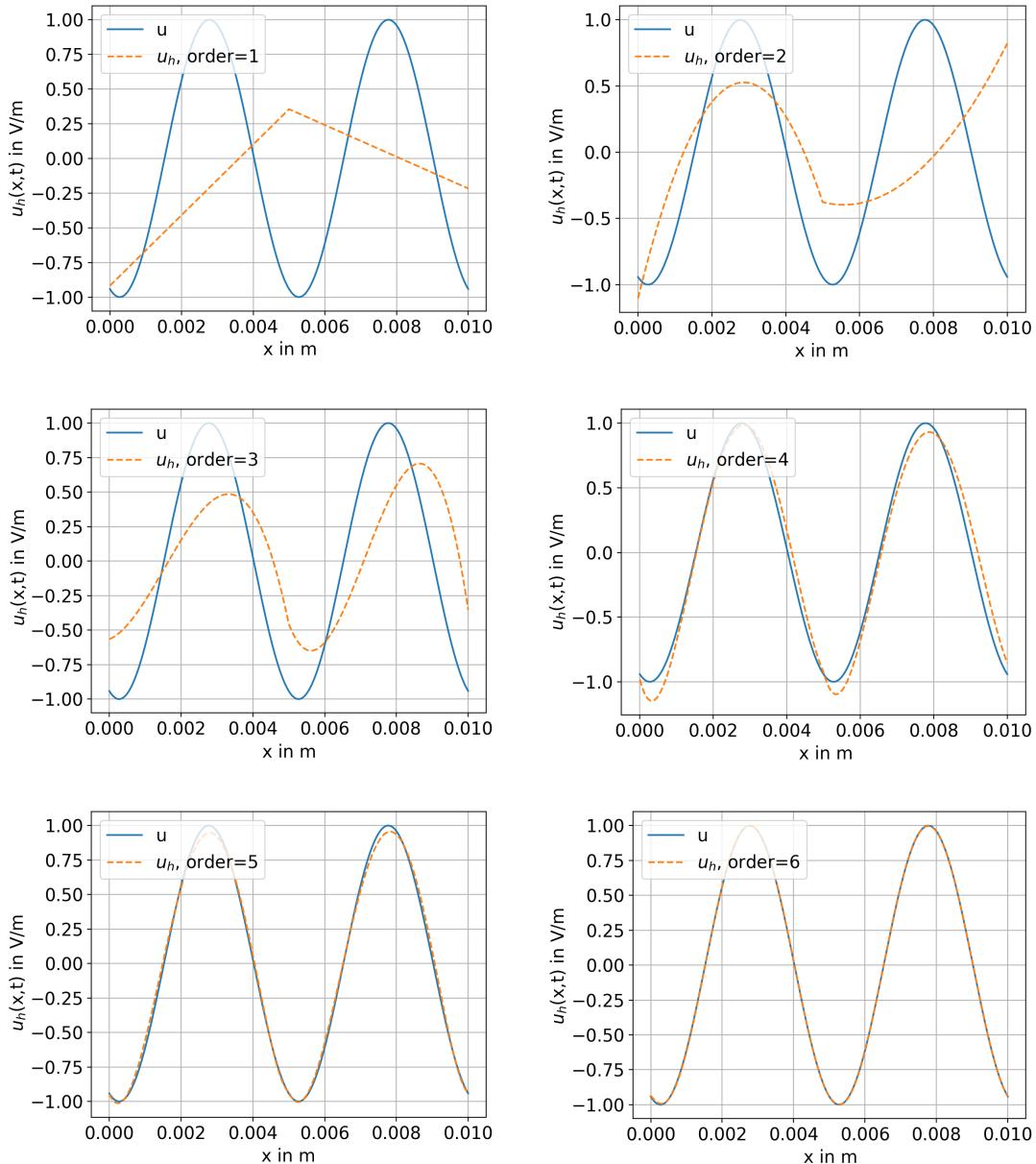


Abbildung 9.3: Lösung  $u_h(x, t)$  für den Zeitschritt  $\frac{m}{2} = 500$  und  $n=2$ .

Die nächste Simulation wird bei  $n=4$  finiten Elementen und  $m=20000$  Zeitschritten (3333 Zeitschritte pro Periodendauer) durchgeführt, wobei der Einfluss der Ordnung der Basisfunktionen untersucht wird. Weiters wird die Simulation für Lagrange- und Legendre-Polynome durchgeführt. Die Berechnung mit einer der beiden unterschiedlichen Polynom-Basen liefert bis zu einer gewissen Ordnung das gleiche Ergebnis. Wobei sich ab ca. der 20-ten Ordnung das Ergebnis für Lagrange-Polynome wieder verschlechtert. Ein weiterer Vorteil der hierarchischen Basis stellt die kürzere Berechnungsdauer dar. Aus diesem Grund werden die meisten der weiteren Simulationen mit den hierarchischen Basisfunktionen durchgeführt.

Die folgenden Fehlerberechnungen der numerischen zur analytischen Lösung werden entweder zeitlich über  $[\lfloor \frac{k}{l}m \rfloor, m]$  mit  $l > k; k, l \in \mathbb{N}$  oder räumlich und zeitlich gemittelt.

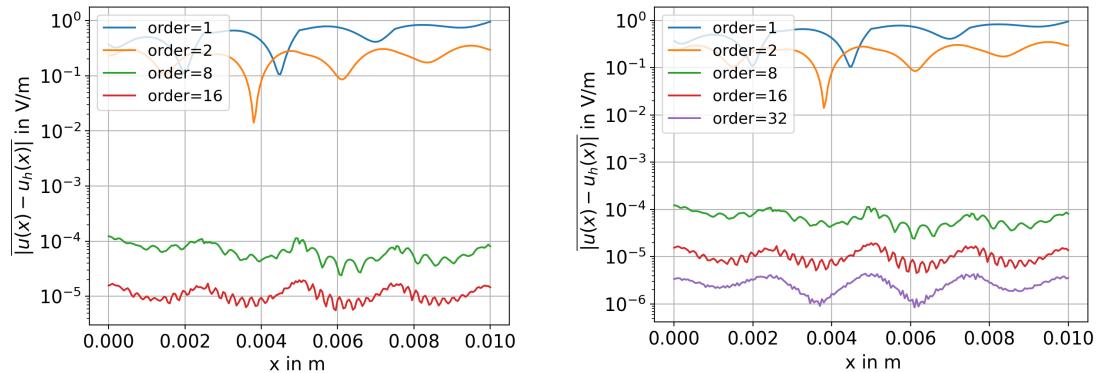


Abbildung 9.4: Fehler der Lösung  $u_h(x, t)$  für  $m=20000$ , zeitlich gemittelt über  $[\lfloor \frac{m}{2} \rfloor, m]$  bei  $n=4$  mit (a) Lagrange- und (b) Legendre-Basisfunktionen.

Für die Berechnung aus Abb. 9.5 wurde gegenüber Abb. 9.4 die Anzahl der FE von  $n=4$  auf  $n=50$  und die Anzahl der Zeitschritte  $m=20000$  auf  $m=40000$  gesteigert. Es zeigt sich wie erwartet ein schnelleres Konvergenzverhalten bei erhöhen der Ordnung der Basisfunktionen. Schon bei der neunten Ordnung ist in Abb. 9.5 ein genaueres Ergebnis erreicht.

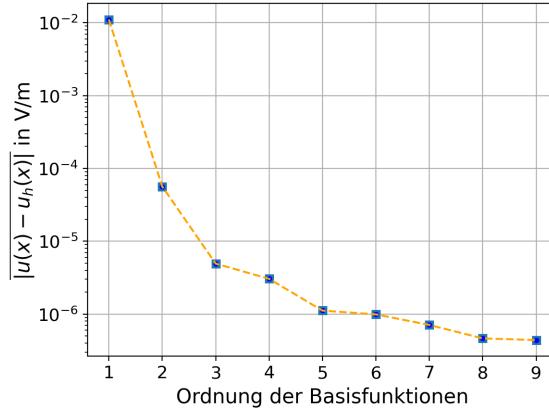


Abbildung 9.5: Fehler der Lösung  $u_h(x, t)$  für  $m=40000$ , zeitlich über  $\left[\left\lfloor \frac{m}{2} \right\rfloor, m\right]$  und räumlich gemittelt bei  $n=50$  mit Legendre-Polynomen.

In der nächsten Simulation, die in Abb. 9.6 dargestellt ist, zeigt sich, dass bei zu geringer Anzahl der Zeitschritte die Lösung deutlich schlechter ausfällt und zu dem rechten Rand hin ungenauere Ergebnisse liefert, siehe Abb. 9.4 und Abb. 9.6. In den weiteren Simulationen wurde darauf geachtet, dass die Berechnungen bei ausreichender Anzahl der Zeitschritte  $m$  durchgeführt wurden. Auch hier ist zwischen Lagrange und Legendre Polynomen kein Unterschied festzustellen.

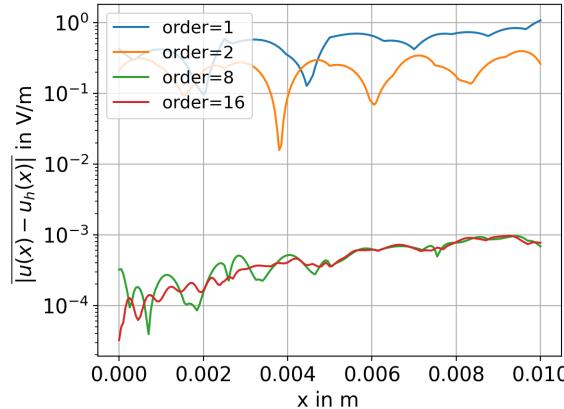


Abbildung 9.6: Fehler der Lösung  $u_h(x, t)$  für  $m=1000$ , zeitlich gemittelt über  $\left[\left\lfloor \frac{m}{2} \right\rfloor, m\right]$  bei  $n=4$  mit Legendre-Basisfunktionen.

In Abb. 9.7 und 9.8 ist das Verhalten der numerischen Berechnung bei Erhöhung der Anzahl der FE dargestellt.

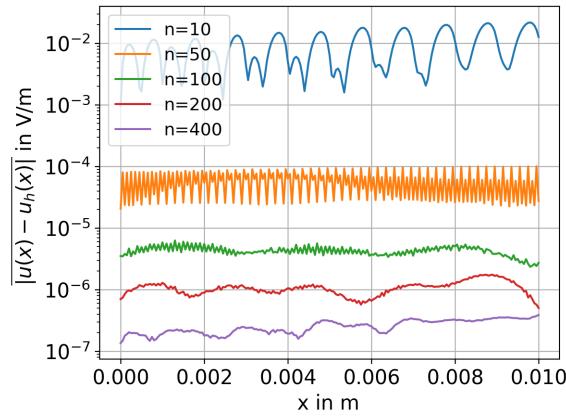


Abbildung 9.7: Fehler der Lösung  $u_h(x, t)$  für  $m=50000$ , zeitlich gemittelt über  $[\lfloor \frac{m}{2} \rfloor, m]$  bei Legendre-Polynomen zweiter Ordnung.

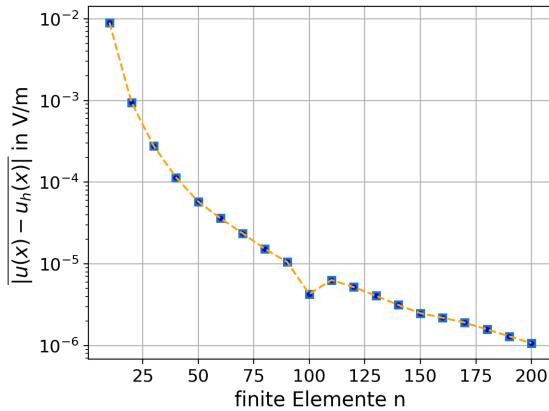


Abbildung 9.8: Fehler der Lösung  $u_h(x, t)$  für  $m=50000$ , zeitlich über  $[\lfloor \frac{m}{2} \rfloor, m]$  und räumlich gemittelt bei Legendre-Polynomen zweiter Ordnung.

Bei den Simulationen hat sich gezeigt, dass eine geringere Anzahl der FE bei höherer Ordnung der Basisfunktionen die bessere Wahl ist als eine höhere Anzahl der FE bei niedriger Ordnung der Basisfunktionen.

## 9.2 Reflexion an einem perfekten Leiter

Im Weiteren wird die Situation betrachtet, bei welcher sich im Gebiet ein perfekter Leiter befindet, der dafür sorgt, dass die gesamte Welle reflektiert wird. Dies kommt daher, dass Metalle mit einer hinreichend großen Leitfähigkeit  $\sigma$  eine sehr gute Reflektivität  $R$  aufweisen, die nahe bei Eins liegt.

In einem idealen Leiter ( $\sigma \rightarrow \infty$ ) kann kein elektrisches Feld existieren<sup>2</sup> und daher auch keine Energie transportiert werden.

Aus den Maxwell-Gleichungen ergibt sich durch die Stetigkeit der Tangentialkomponente des elektrischen Feldes  $\mathbf{n} \times [\mathbf{E}] = \mathbf{0}$  und der Tatsache, dass bei einer sehr hohen Leitfähigkeit das elektrische Feld verschwinden muss, dass die einfallende Welle superponiert mit dem reflektierten Feld beim Grenzübergang Null ergeben muss. Es folgt die Situation bei welcher die Welle am linken Rand angeregt wird und bis zum perfekten Leiter läuft, bei dem sie dann komplett reflektiert wird und eine Phasenverschiebung von  $\pi$  erfährt. Dabei bleibt die gesamte Amplitude nach der Reflexion erhalten. In der analytischen Berechnung wird gezeigt, dass sich für diese Feldsituation eine stehende Welle ergibt.

### 9.2.1 Analytische Beschreibung

Für die analytische Berechnung wird zu Beginn eine Koordinatentransformation durchgeführt, sodass die Superposition der hinlaufenden und reflektierten Welle einfacher zu behandeln ist.

Das neue Koordinatensystem ist so gewählt, dass der neue Ursprung an der Grenzfläche zum Leiter liegt. Es ergibt sich die Wellenzahl der Reflektierten  $k_r$  (engl. reflected) aus der Wellenzahl der einfallenden Welle  $k_i$  (engl. incident)  $k = k_i, k_r = -k$ .

Die Größe  $x$  wird so transformiert, sodass zum Zeitpunkt  $t = t_i$  am Ort  $x = d_i$  der Reflexionsvorgang beginnt, sprich die Welle beim Leiter angekommen ist.<sup>3</sup>

$$x' = x - d_i \quad (9.12)$$

Die einfallende und reflektierte Welle lassen sich darstellen durch:

$$\tilde{\mathbf{E}}_i(x', t) = \tilde{\mathbf{E}}_{i,0} e^{-j(kx' - \omega t)} \quad (9.13)$$

$$\tilde{\mathbf{E}}_r(x', t) = \tilde{\mathbf{E}}_{r,0} e^{-j(-kx' - \omega t)} \quad (9.14)$$

---

<sup>2</sup>Aus dem lokalen ohmschen Gesetz (3.7) folgt: Für eine endliche Stromdichte  $\mathbf{J}$  muss bei einer Leitfähigkeit die gegen unendlich geht, das elektrische Feld  $\mathbf{E}$  verschwinden.

<sup>3</sup>In Python muss um die analytische Lösung korrekt darzustellen, die Anfangsphasenlage und die sich durch die Koordinatentransformation (9.12) ergebende Phasenverschiebung berücksichtigt werden.

Aufgrund der perfekten Reflexion ergibt sich nach der Zeit  $t = 2t_i$  die Gesamtfeldstärke vor der Grenzschicht  $x' < 0$  zu einer stehenden Welle. Diese besitzt Knotenpunkte, bei welchen für alle Zeiten das Feld verschwindet, siehe Gleichung (9.20). Nach der Grenzschicht für  $x' > 0$ , wie oben erwähnt, verschwindet das elektrische Feld.

Die Bedingung für die komplexen Amplitudenvektoren lautet:<sup>4</sup>

$$\tilde{\mathbf{E}}_{total}(0, \omega) = \tilde{\mathbf{E}}_{i,0} + \tilde{\mathbf{E}}_{r,0} = 0 \quad (9.15)$$

Für  $t > 2t_i$  gilt:

$$\tilde{\mathbf{E}}_{total}(x', \omega) = \tilde{\mathbf{E}}_{i,0}e^{-j k x'} + \tilde{\mathbf{E}}_{r,0}e^{j k x'} \quad (9.16)$$

$$\tilde{\mathbf{E}}_{i,0} = -\tilde{\mathbf{E}}_{r,0} = \tilde{\mathbf{E}}_0 \quad (9.17)$$

$$\tilde{\mathbf{E}}_{total}(x', \omega) = -\tilde{\mathbf{E}}_0(e^{j k x'} - e^{-j k x'}) \quad (9.18)$$

$$= -2j\tilde{\mathbf{E}}_0 \sin(kx') \quad (9.19)$$

$$= -2j\mathbf{E}_0 e^{j\varphi} \sin(kx') \quad (9.20)$$

Für die Betrachtung im Zeitbereich für Zeiten  $t > 2t_i$  wird der Realteil von dem Produkt des zeitabhängigen Phasenterms und des komplexen Amplitudenvektors gebildet:

$$\mathbf{E}_{total}(x', t) = \operatorname{Re}\{\tilde{\mathbf{E}}_{total}(x', \omega)e^{j\omega t}\} \quad (9.21)$$

$$= -2\mathbf{E}_0 \sin(kx') \operatorname{Re}\{je^{j(\omega t + \varphi)}\} \quad (9.22)$$

$$= 2\mathbf{E}_0 \sin(kx') \sin(\omega t + \varphi) \quad (9.23)$$

Das erhaltene Ergebnis für die harmonische Anregung kann gut mittels der Animation der numerischen Lösung beobachtet werden. Dafür muss der Python Code aus 10.1 ausgeführt werden, für den die passenden Materialparameter in 10.2 vorzugeben sind.

## 9.2.2 Numerische Berechnung

Der Einschwingvorgang für die Reflexion an einem perfekten Leiter ist in Abb. 9.9 zu sehen. Dabei sieht man in Abb. 9.9 (a), dass die elektrische Leitfähigkeit  $\sigma(x)$  bei  $x = \frac{3(b-a)}{4}$  von Null auf  $10^{16} Sm^{-1}$  springt. Abb. 9.9 (d) zeigt den eingeschwungenen Zustand, welcher eine stehende Welle darstellt. Da für die folgenden Fehlerberechnungen nur der eingeschwungene Zustand untersucht wird, wird nur die Lösung für die Zeitschritte im Intervall  $[\lfloor \frac{3m}{4} \rfloor, m]$  betrachtet.

---

<sup>4</sup>Der Phasenterm  $e^{j\varphi}$  tritt aufgrund der Phasenverschiebung auf.

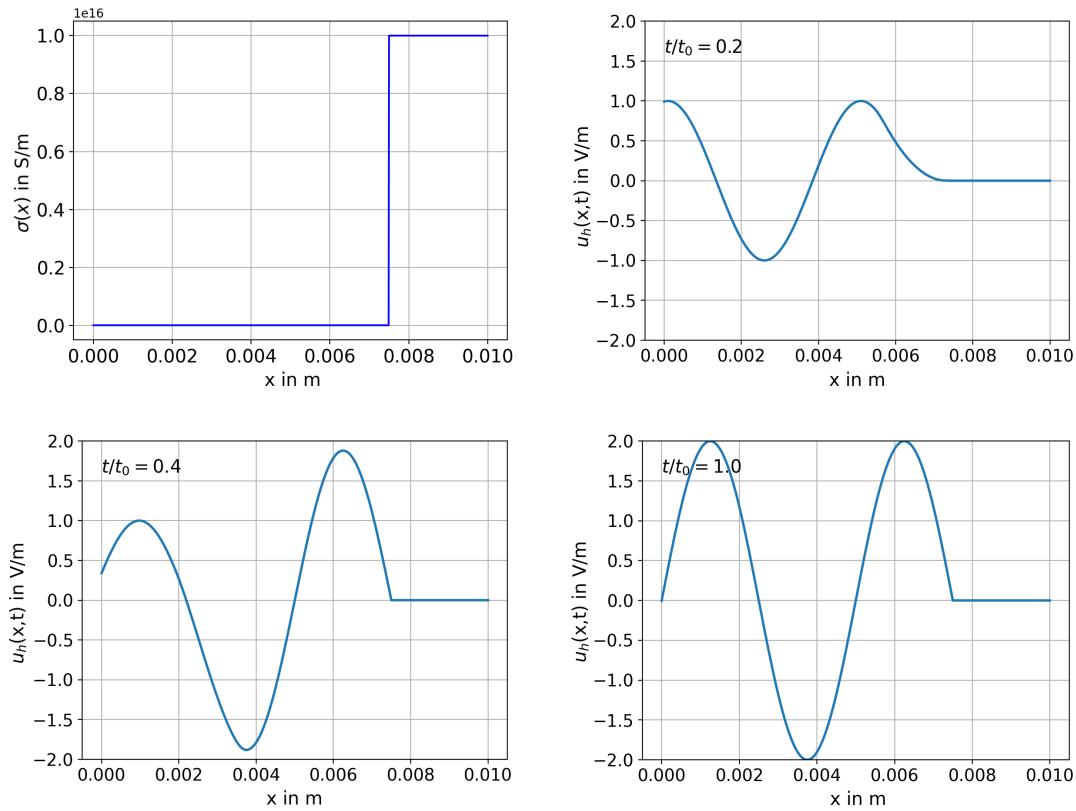


Abbildung 9.9: (a) Elektrische Leitfähigkeit  $\sigma(x)$ . (b)-(d) Einschwingvorgang der stehenden Welle.

In Abb. 9.10 ist die Näherungslösung  $u_h(x,t)$  für die in der Legende angegebenen Zeitpunkte  $t/t_0$  ausgewertet. Es handelt sich dabei um den eingeschwungenen Zustand. Man erkennt hierbei gut die Knotenpunkte der stehenden Welle, für welche das elektrische Feld zu jedem Zeitpunkt verschwindet. Auch gut zu erkennen ist, wie die ortsfeste Lage der Welle mit dem Term  $\sin(\omega t + \varphi)$  schwingt.

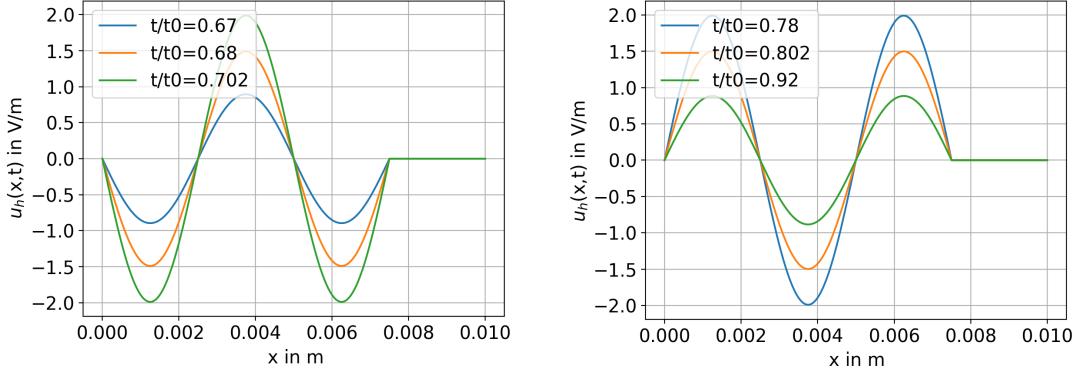


Abbildung 9.10: Näherungslösung  $u_h(x, t)$  im eingeschwungenen Zustand.

Abb. 9.11 bzw. 9.12 behandeln das Konvergenzverhalten bei Erhöhung der Ordnung der Basisfunktionen bzw. der Anzahl der finiten Elemente. Es zeigt sich wie auch schon bei den Simulationen aus 9.1.2, dass das Erhöhen der Ordnung der Basisfunktionen sinnvoller ist als das Gebiet feiner zu diskretisieren. Weiters ist darauf hinzuweisen, dass nur der Bereich links vom perfekten Leiter untersucht wird, da das elektrische Feld rechts davon vollständig verschwindet.

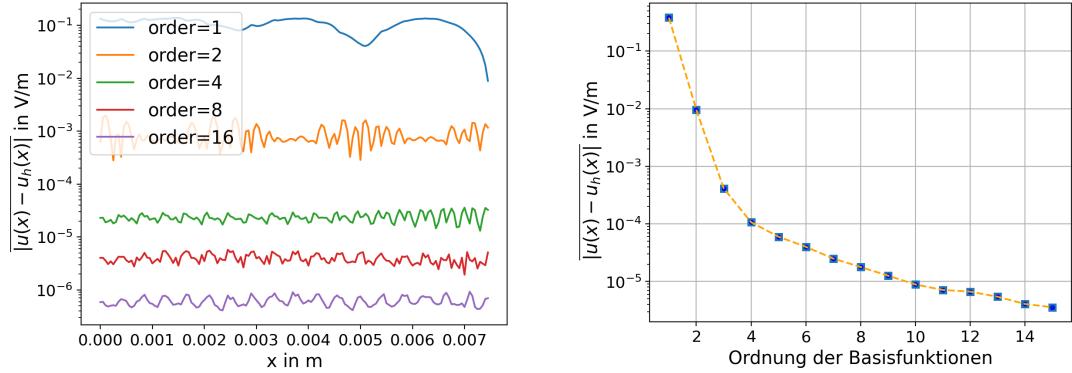


Abbildung 9.11: Fehler der Lösung  $u_h(x, t)$  für (a)  $m=60000$ , zeitlich gemittelt über  $\left[ \lfloor \frac{3m}{4} \rfloor, m \right]$  bei  $n=12$  mit Legendre-Basisfunktionen höherer Ordnung. (b) Konvergenzverhalten bei zusätzlicher räumlicher Mittelung.

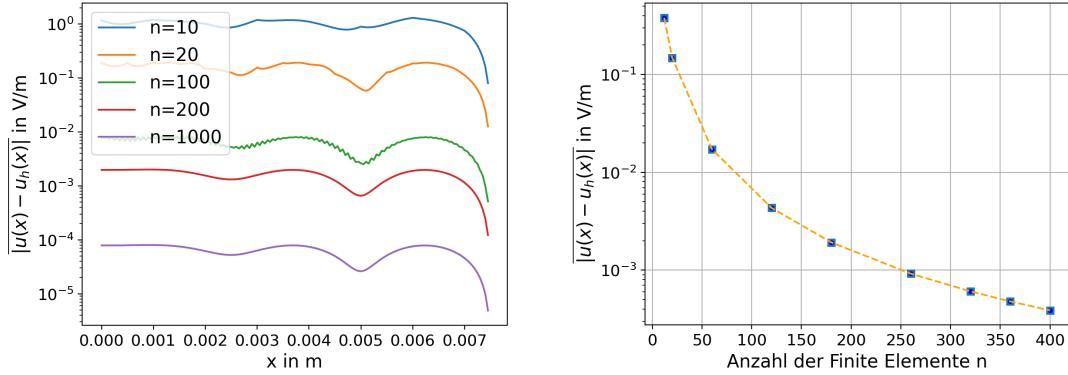


Abbildung 9.12: Fehler der Lösung  $u_h(x, t)$  für (a)  $m=60000$ , zeitlich gemittelt über  $\left[\lfloor \frac{3m}{4} \rfloor, m\right]$  bei linearen Basisfunktionen. (b) Konvergenzverhalten bei zusätzlicher räumlicher Mittelung.

Für Abb. 9.13 wurde die Intervallanzahl bewusst teilweise unpassend gewählt, sodass für die ein  $n$ , welches nicht durch 4 teilbar ist, die Berechnung ein ungenauereres Ergebnis liefert, da die Intervallgrenze beim perfekter Leiter nicht mehr bei  $x = \frac{3(b-a)}{4}$  liegt.

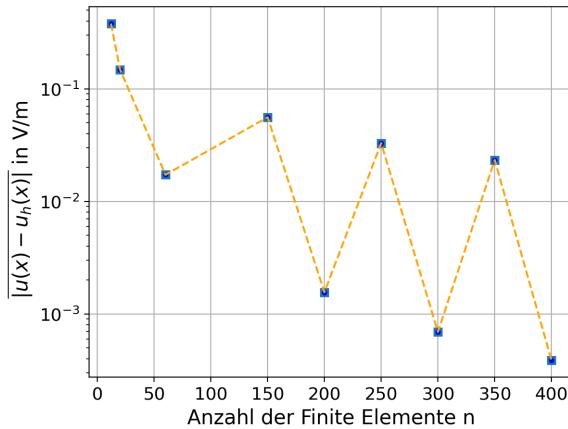


Abbildung 9.13: Konvergenzverhalten wie in Abb. 9.12 (b), jedoch bei unterschiedlicher Anzahl der FE.

### 9.3 Übergang der Welle zwischen zwei Medien unterschiedlicher Permittivität

In dem nächsten Simulationsbeispiel wird das Problemgebiet in zwei Halbräume geteilt, die aus unterschiedlichen dielektrischen Materialien bestehen. Die Grenzschicht wird dabei im letzten Viertel  $x = \frac{3(b-a)}{4}$  des Problemgebiets  $[a,b]$  gewählt, mit der Permittivität  $\epsilon(x)$ , die bei dem Übergang der beiden Materialien springt.

Zur einfachen analytischen Berechnung wird der Übergang vom leeren Raum mit der relativen Permittivität  $\epsilon_r = 1$  in ein Dielektrikum mit  $\epsilon_r = 9$  gewählt. Daraus folgt, wie bei der analytischen Berechnung gezeigt wird, der Fall, dass eine Welle mit genau der halben Amplitude reflektiert und transmittiert wird.

#### 9.3.1 Analytische Berechnung

Für die analytische Berechnung wird dabei eine einfallende, eine transmittierte und eine reflektierte Welle angesetzt, wobei es sinnvoll ist die Koordinatentransformation aus (9.12) zu nutzen.

Dabei ist das Ausbreitungsverhalten in den beiden Medien durch die Dispersionsrelation aus der Wellengleichung bekannt. Da sich die Welle mit der um den Brechungsindex  $n = \sqrt{\epsilon_r}$  reduzierten Vakuum-Lichtgeschwindigkeit im Medium ausbreitet und die Frequenz konstant bleibt, ändert sich somit die Wellenzahl  $k = \frac{\omega}{c}$  und damit auch die Wellenlänge  $\lambda = \frac{2\pi}{k}$ .

Aus den Randbedingungen für die Stetigkeit der elektrischen und magnetischen Feldstärke folgt für eine ebene Welle mit senkrechter Polarisierung zur Einfallsebene der Reflexions- und Transmissionskoeffizient nach Fresnel. Diese geben an, welche Amplitude die reflektierte bzw. transmittierte Welle besitzt (vgl. [Rei12], S.41).

Für den Einfall der Welle auf die Grenzfläche (Normaleinfall) erhält man für die Fresnel-Koeffizienten:

$$\begin{aligned}\varrho_r &= \frac{n_i - n_t}{n_i + n_t} \\ \varrho_t &= \frac{2n_i}{n_i + n_t}\end{aligned}\tag{9.24}$$

Bei den Brechungsindizes  $n_i = 1$  und  $n_t = 3$  ergeben sich die Fresnel-Koeffizienten zu  $\varrho_r = -1/2$  und  $\varrho_t = 1/2$  (siehe (9.24)). Da sich die analytische Lösung der Wellenausbreitung aus den analytischen Lösungen aus den Kapiteln 9.1 und 9.2 ableiten lässt, wird die Berechnung hier nur kurz beschrieben.

Im eingeschwungenen Zustand sieht die Situation so aus, dass der bei  $x = \frac{3(b-a)}{4}$  reflektierte Teil der Welle mit der Amplitude  $\varrho_r = -1/2$ , überlagert mit der halben Amplitude der von

### 9.3. Übergang der Welle zwischen zwei Medien unterschiedlicher Permittivität

links zum zweiten Medium hinlaufenden Welle eine stehende Welle bildet, welche schon in 9.2 beschrieben ist. Der noch nicht berücksichtigte Teil der Welle mit der Amplitude  $1/2$  läuft vom linken Rand kommend Richtung Grenzschicht und wird der stehenden Welle überlagert. Für das rechts von der Grenzschicht  $x = \frac{3(b-a)}{4}$  liegende Gebiet muss man den transmittierten Teil darstellen, welcher mit der Amplitude  $\varrho_t = 1/2$ , Wellenlänge  $\lambda = \frac{\lambda_0}{n_t}$  und mit der um  $n_t = \sqrt{\varepsilon_r} = 3$  reduzierten Vakuum-Lichtgeschwindigkeit  $c_0$  in positive x-Richtung aus dem Problemgebiet bewegt. Für die Berechnung in Python wurde dafür die analytische Lösung aus 9.1 und 9.2 mit jeweils  $1/2$  multipliziert und überlagert. Die Ausbreitungssituation für den rechten Bereich wurde dafür nachträglich überschrieben, wobei man hier auf die richtige Phasenlage  $\varphi$  und Wellenzahl  $k = k_0 n_t$  achten muss.

#### 9.3.2 Numerische Berechnung

In den Abb. 9.14 ist der Einschwingvorgang für den in der analytischen Berechnung beschriebenen Ausbreitungssituation der elektromagnetischen Welle dargestellt.

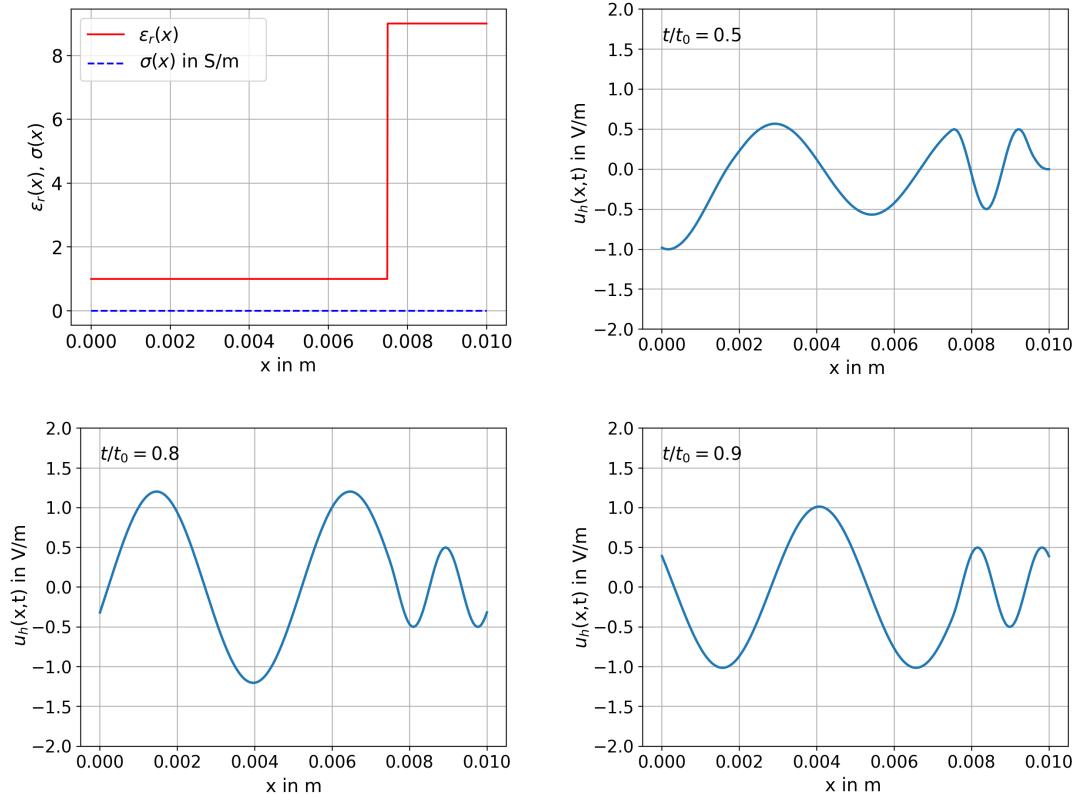


Abbildung 9.14: (a) Relative Permittivität  $\varepsilon_r(x)$  und elektrische Leitfähigkeit  $\sigma(x)$ . (b)-(d) Einschwingvorgang der Ausbreitungssituation.

## 9. VERGLEICH DER NÄHERUNGSLÖSUNG MIT DER ANALYTISCHEN LÖSUNG

---

In den Abb. 9.15 und 9.16 ist wie auch in den Kapiteln 9.1 und 9.2 das Konvergenzverhalten für Erhöhung der Ordnung der Basisfunktionen bzw. der Anzahl der finiten Elemente dargestellt.

Wie bereits besprochen führt das Erhöhen der Ordnungen der Basisfunktionen statt das Problemgebiet feiner zu diskretisieren zu einer effizienteren Berechnung.

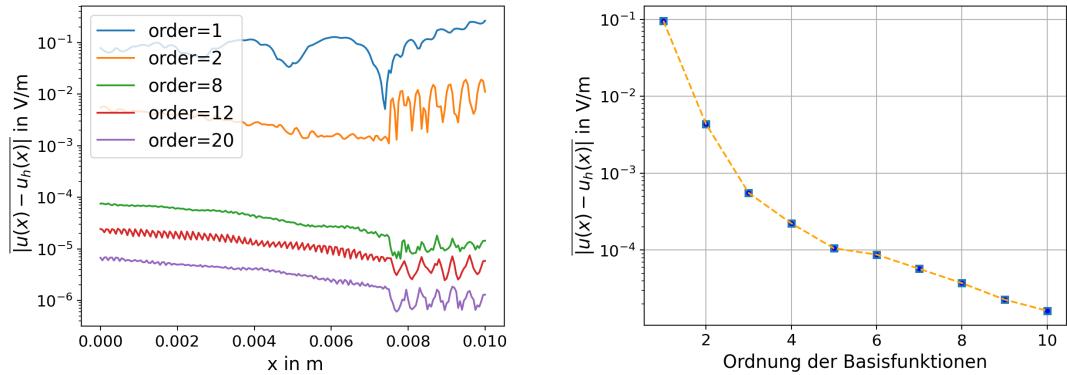


Abbildung 9.15: Fehler der Lösung  $u_h(x, t)$  für (a)  $m=100000$ , zeitlich gemittelt über  $[\lceil \frac{3m}{4} \rceil, m]$  bei  $n=24$  mit Legendre-Basisfunktionen höherer Ordnung. (b) Konvergenzverhalten bei zusätzlicher räumlicher Mittelung.

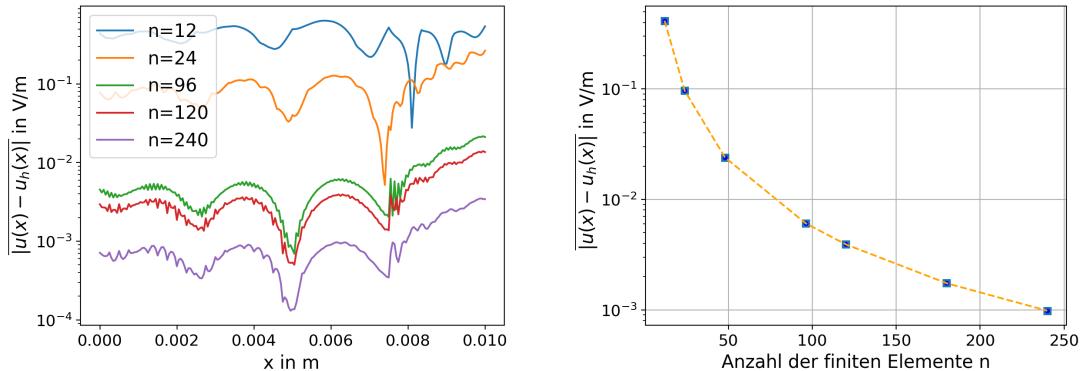


Abbildung 9.16: Fehler der Lösung  $u_h(x, t)$  für (a)  $m=50000$ , zeitlich gemittelt über  $[\lceil \frac{3m}{4} \rceil, m]$  bei linearen Basisfunktionen. (b) Konvergenzverhalten bei zusätzlicher räumlicher Mittelung.

## 9.4 Weitere Berechnungen

In der Simulation aus Abb. 9.17 wird auf die Absorption des Mediums bei einer elektrischen Leitfähigkeit, die konstant über das gesamte Problemgebiet den Wert  $\sigma(x) = 1 \text{ Sm}^{-1}$  annimmt. Dabei wird die analytische Lösung  $u(x, t)$  mit der numerischen Näherungslösung  $u_h(x, t)$  verglichen und der absolute Fehler dargestellt. Für die analytische Lösung ergibt sich

$$\tilde{\mathbf{E}}(x, \omega) = \tilde{\mathbf{E}}_{i,0} e^{-j\tilde{k}x} \quad (9.25)$$

$$= \tilde{\mathbf{E}}_{i,0} e^{-j\operatorname{Re}\{\tilde{k}\}x} e^{-\operatorname{Im}\{-\tilde{k}\}x} \quad (9.26)$$

mit dem komplexen Wellenvektor  $\tilde{k} = \sqrt{\varepsilon(x) - j\frac{\sigma(x)}{\omega\varepsilon(x)}}$ , welcher aus der Dispersionsrelation folgt und in ([Rei12], S.55) beschrieben und hergeleitet wird.

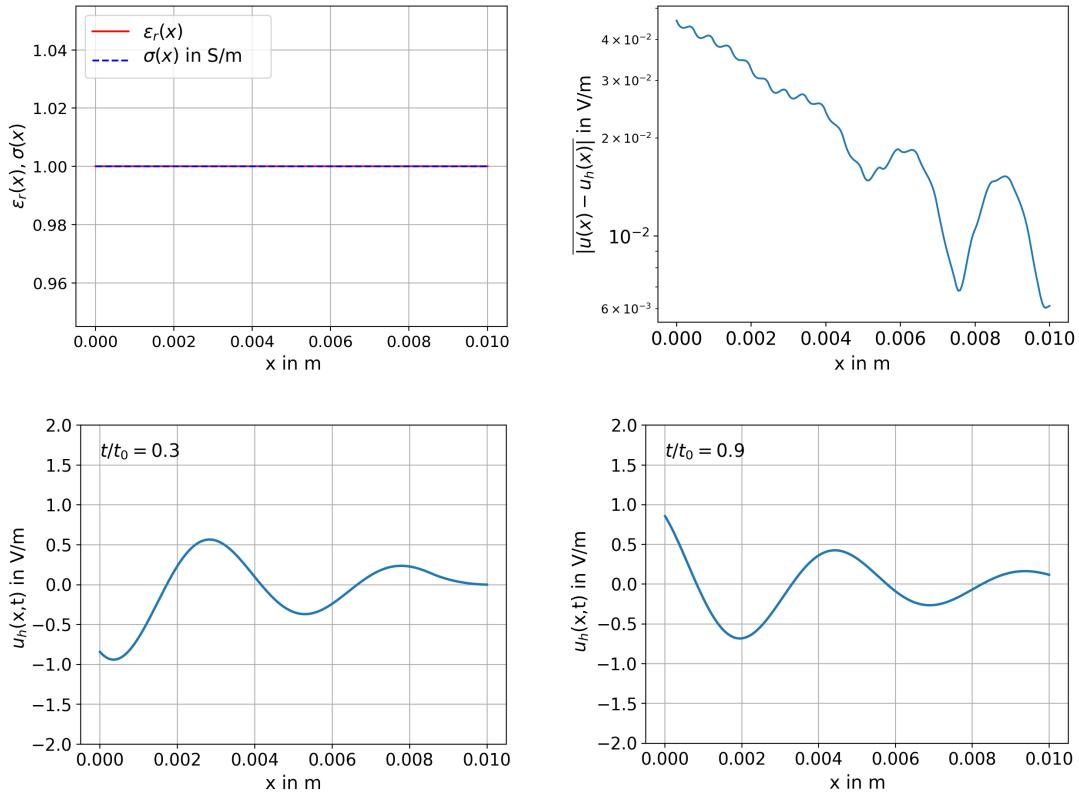


Abbildung 9.17: (a) Materialparameter. (b) Fehler der Lösung  $u_h(x, t)$  bei  $n=20$ ,  $m=1000$  mit Legendre-Basisfunktionen zweiter Ordnung, zeitlich gemittelt über  $[\lfloor \frac{7m}{8} \rfloor, m]$ . (c)-(d) Simulation der gedämpften Welle.

Genauere Ergebnisse als in Abb. 9.17 zu sehen sind, wurden auch nicht mit einer feineren Diskretisierung oder mit Basisfunktionen höherer Ordnung erreicht. Das liegt daran, dass die beiden Randbedingungen für eine Leitfähigkeit, die am jeweiligen Rand  $x=a$  oder  $x=b$  einen Wert ungleich Null besitzt, dafür sorgt, dass die Faktorisierung der Wellengleichung für die Randbedingung nicht exakt ist (vgl. [Arn07]).

Um ein Konvergenzverhalten zu erhalten, wie es in Kapitel 9.1-9.3 zu sehen ist, sollte man die elektrische Leitfähigkeit  $\sigma(x)$  an beiden Rändern  $x=a$  und  $x=b$  gegen Null gehen lassen. Dafür ergibt sich eine exakte Faktorisierung der Wellengleichung.

In den beiden nachfolgenden Simulationen soll der Vorteil der numerischen Simulation für kompliziertere Materialverläufe sichtbar werden. Hierbei ist die Angabe einer analytischen Lösung für die Situation aus Abb. 9.18 nicht mehr so einfach möglich.

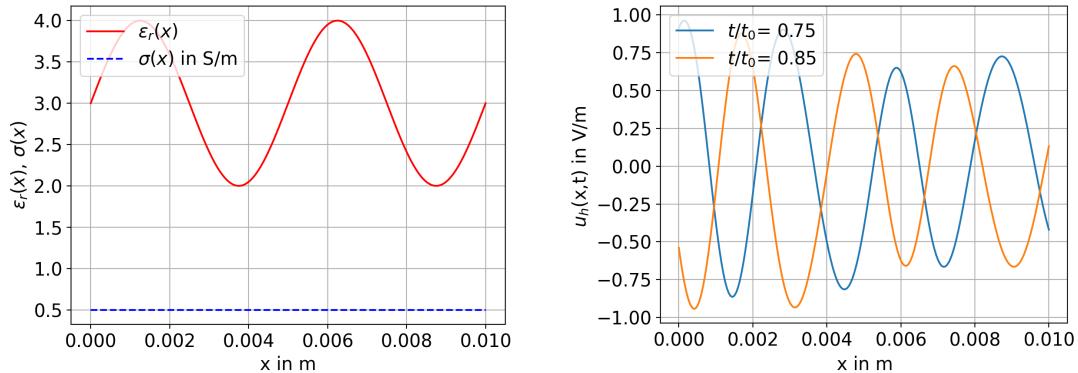


Abbildung 9.18: Kompliziertere Vorgabe der Materialparameter  $\epsilon(x)$  und  $\sigma(x)$  und dazugehörige Lösungen.

In Abb. 9.19 ist ein dispersives Material behandelt, welches bei zwei Kreisfrequenzen  $\omega_1$  und  $\omega_2 = 1.2\omega_1$  untersucht wird. Für diesen Fall ist zuerst die Simulation für  $\omega_1$  und  $\omega_2$  einzeln durchgeführt und darunter die überlagerte Lösung dargestellt. Die Gesamtlösung erhält man durch Addition der Einzellösungsmatrizen.

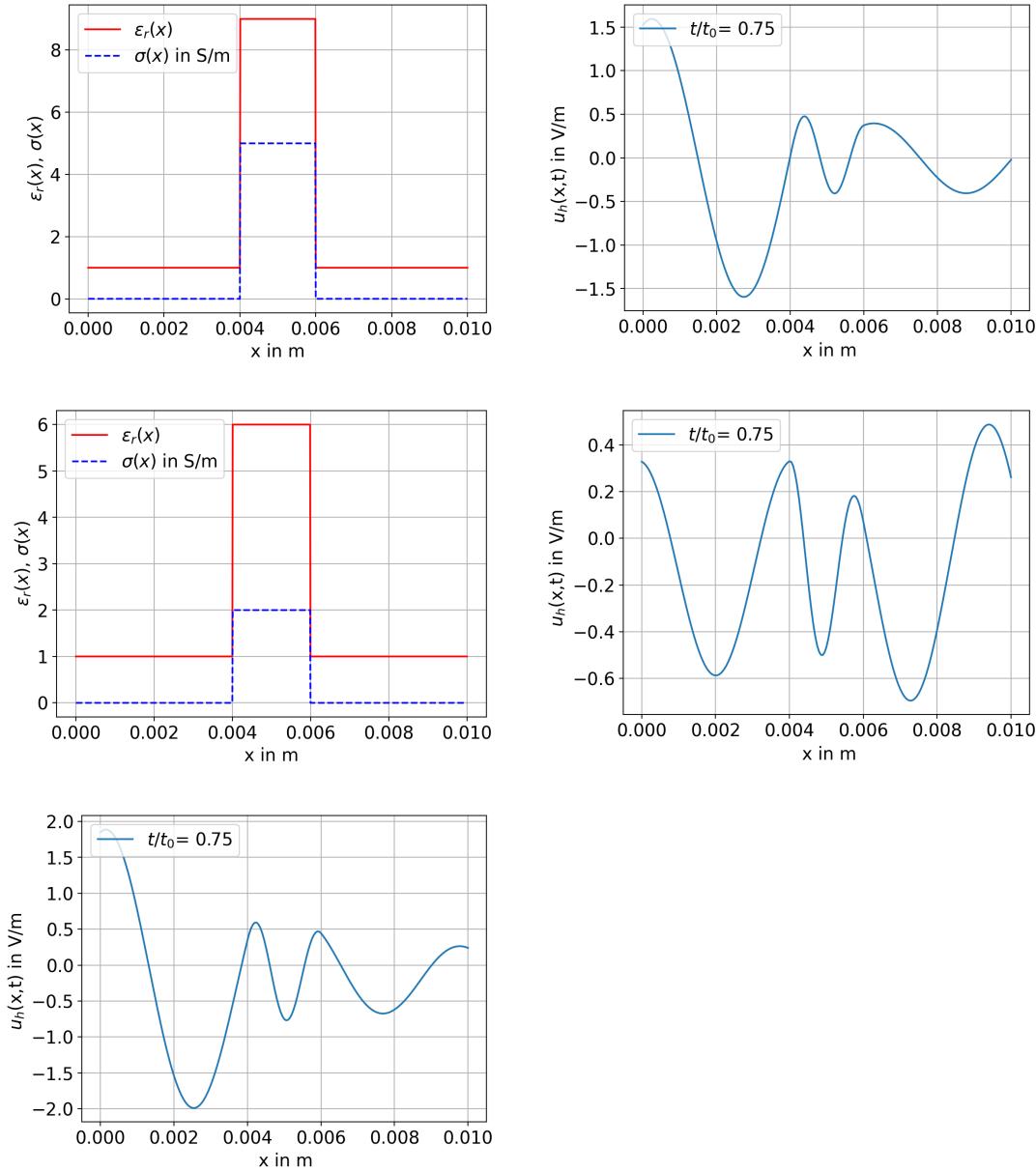


Abbildung 9.19: Frequenzabhängiges Material bei Ausbreitung zweier Wellen mit den Kreisfrequenzen  $\omega_1$  (a) und  $\omega_2$  (c). (b) Lösung  $u_h(x, t)$  für  $\omega_1$ . (d) Lösung  $u_h(x, t)$  für  $\omega_2$ . (e) Gesamtlösung  $u_h(x, t)$  für  $\omega_1$  und  $\omega_2$ .



# 10

## KAPITEL

# Python Source-Code

Um die Simulation durchführen zu können, müssen sich alle Python-Module 10.1-10.7 in demselben Ordner befinden. Dabei müssen mit dem Modul *settings\_domain1D.py* die Materialparameter der Permittivität  $\varepsilon(x)$  und der elektrischen Leitfähigkeit  $\sigma(x)$  für das Problemgebiet  $[a,b]$  als Funktionen vorgegeben werden. Im Modul *test.py* können Simulationsparameter wie die Anzahl der finiten Elemente, Anzahl der Zeitschritte und die Ordnung der Basisfunktionen angegeben werden. Nachdem alle Parameter festgelegt sind, kann die Simulation durch Ausführen des Moduls *test.py* durchgeführt werden. Dabei werden zuerst die Materialparameter angezeigt, bevor die Simulation startet.

Die Berechnung ist in Python in folgende Module unterteilt:

- *test.py*
- *settings\_domain1D.py*
- *main\_FEM.py*
- *integration.py*
- *timeIntegrationMethod.py*
- *shapeFunctions.py*
- *plotResult.py*

### 10.1 test.py

```
import numpy as np
from settings_domain1D import *
from matplotlib import pyplot as plt
```

```
import shapeFunctions
import main_FEM
import plotResult

# Zeitbereich [s]
t_0 = 1e-10
c_0 = 299792458
# Kreisfrequenz [Hz]
omega = 4 * np.pi * c_0 / b_ #genau zwei Wellenlaengen im Problemgebiet

#Damit Anfangswert und Anfangsgeschwindigkeit Null sind
#bessere Ergebnisse fuer Zeitschrittverfahren
t_phi= 2/omega
a= 1/np.sqrt(2)/t_phi**2
phi= np.pi/4-omega*t_phi

# Anregung
def w(t):
    if(t<t_phi):
        return a*t**2
    return np.sin(omega * t+phi)

# Ableitung der Anregung
def w_der(t):
    if (t < t_phi):
        return 2*a * t
    return np.cos(omega * t+phi) * omega

'''

Funktion_eps(x) und sigma(x) darstellen
'''

if (1):
    x = [b_ / 1000 * i for i in range(0, 1001)]
    y_eps = [eps(xi) / eps0 for xi in x]
    y_sigma = [sigma(xi) for xi in x]

    fig = plt.figure()
    ax1 = fig.add_subplot(111)
    ax1.plot(x, y_eps, 'r')
    ax1.set_ylabel('$\epsilon_r(x)$', fontsize= 15)
    ax1.set_xlabel('$x[m]$', fontsize=15)

    ax2 = ax1.twinx()
```

```

ax2.plot(x, y_sigma, 'b--')
ax2.set_ylabel('$\sigma(x) \cdot [(\Omega_m)^{-1}]$', color='b', fontsize= 15)
for tl in ax2.get_yticklabels():
    tl.set_color('b')
fig.suptitle('Materialparameter $\epsilon_r(x)$, $\sigma(x)$', fontsize=15)
ax1.tick_params(axis="x", labelsize=15)
ax1.tick_params(axis="y", labelsize=15)
ax2.tick_params(axis="y", labelsize=15)
plt.show()

'''
Anregung_w(t) darstellen
'''

if (1):
    x = [t_0 / 1000 * i for i in range(0, 1001)]
    y = [w(xi) for xi in x]
    plt.plot(x, y)
    plt.xlabel('t[s]', fontsize= 15)
    plt.ylabel('w(t)[V/m]', fontsize= 15)
    plt.xticks(fontsize= 15)
    plt.yticks(fontsize=15)
    plt.title('Anregung_w(t)(linker Rand)', fontsize=15)
    plt.grid()
    plt.show()

#Basisfunktionen(Shape Functions)
SF = 1 #0->Lagrange, 1->Legendre
m = 1000 #Anzahl der Zeitschritte
nFE= 50 #Anzahl der finiten Elemente
nSF = 3 #Ordnung der Basisfunktionen
integrationOrderM = nSF + 1
integrationOrderC = nSF + 1
n = nFE * nSF + 1
h_t = t_0 / m
h_x = (b_- - a_-) / nFE

'''
Animation

```

```
'''  
if (1):  
    U = np.zeros(shape=(n, m))  
    U = main_FEM.getU_main(a_, b_, nFE, n, m, nSF, h_t, h_x,  
                           integrationOrderM, integrationOrderC, w, w_der, SF)  
    plotResult.animation2D(U, m, nSF, h_x, nFE, SF)
```

## 10.2 settings\_domain1D.py

```
#Problemgebiet [m]  
a_= 0  
b_= 0.01  
  
mue0= 1.25663706212*1e-6 #magn. Feldkonstante  
eps0= 8.8541878128*1e-12 #elektr. Feldkonstante  
  
#Permittivitaet  
def eps(x):  
    if x> b_*2/5 and x<b_*3/5:  
        return 9*eps0  
    return 1*eps0  
  
#Leitfaehigkeit  
def sigma(x):  
    if x> b_*4/5:  
        return 1e16  
    else:  
        return 0
```

## 10.3 main\_FEM.py

```
import time  
import numpy as np  
import timeIntegrationMethod  
import integration  
  
'''  
FEM + NEWMARK-BETA-VERFAHREN  
Rueckgabe: U  
Spaltenvektoren von U enthalten  
die Koeffizienten der Basisfunktionen fuer einen Zeitschrittdef getU_main(a_, b_, nFE, n, m, nSF, h_t, h_x, integrationOrderM,
```

```

integrationOrderC, w, w_der, SF) :

'''LAUFZEIT'''
startTime = time.time()
'''

DEFINITION DER MATRIZEN UND VEKTOREN
nFE...Anzahl der Finiten Elemente
nSF...Ordnung der Basisfunktionen (Shape functions)
n= nFE*nSF+1
(n)... Anzahl aller Basisfunktionen
'''


#System-Matrizen
K = np.zeros(shape=(n, n)) #Steifigkeits-Matrix
M = np.zeros(shape=(n, n)) #Massenmatrix
C = np.zeros(shape=(n, n)) #Daempfungsmatrix

#Loesungsvektoren
U = np.zeros(shape=(n, m)) #elektrische Feldstaerke
V = np.zeros(shape=(n, m)) #Geschwindigkeit
A = np.zeros(shape=(n, m)) #Beschleunigung

#Lokale Matrix
A_local = np.zeros(shape=(nSF+1, nSF+1)) #lokale Matrix

'''


FINITE ELEMENTE METHODE:
Eintraege der Matrizen K,M und C werden berechnet:
Dabei werden die lokalen Matritzen A fuer jedes Element k der nFE
Finiten Elemente berechnet und anschliessend in
die globale Matrix (K, M, C) eingetragen.

Parameter:
-k-tes Element (wird fuer Transformation benoetigt, eps(x), sigma(x))
-{0,1} es wird ueber die normalen (abgeleiteten) Basisfunktionen integriert
-Gewichte der Gauss-Legendre Quadraturregel
-Gewichte der Gauss-Legendre Quadraturregel
-Funktion (1, mue_esp(x), sig_eps(x)) welche im Integraden vorkommt
'''


'''


STEIFIGKEITSMATRIX K

```

```
'''  
gaussQuadK=integration.GaussLegendreQuadratureRule(nSF+1)  
  
for k in range(0, nFE): #Finite Elemente  
    integration.calculateLocalMatrix(k, 1, gaussQuadK[0], gaussQuadK[1]  
        A_local, integration.Eins, h_x, nSF, SF)  
    integration.local_global_table(k, A_local, K, nSF)  
  
'''  
MASSENMATRIX M  
'''  
gaussQuadM=integration.GaussLegendreQuadratureRule(integrationOrderM)  
  
for k in range(0, nFE):#Finite Elemente  
    integration.calculateLocalMatrix(k, 0, gaussQuadM[0],  
        gaussQuadM[1], A_local, integration.mue_eps, h_x, nSF, SF)  
    integration.local_global_table(k, A_local, M, nSF)  
  
'''  
Daempfungsmatrix  
'''  
gaussQuadC=integration.GaussLegendreQuadratureRule(integrationOrderC)  
  
for k in range(0, nFE):#Finite Elemente  
    integration.calculateLocalMatrix(k, 0, gaussQuadC[0],  
        gaussQuadC[1], A_local, integration.mue_sigma, h_x, nSF, SF)  
    integration.local_global_table(k, A_local, C, nSF)  
  
#Rechter Rand, linker Rand (nichtreflektierende Randbedingungen)  
C[0][0]= C[0][0]+ np.sqrt(integration.mue_eps(a_))  
C[n-1][n-1]= C[n-1][n-1]+ np.sqrt(integration.mue_eps(b_))  
  
print('Laufzeit_(K,M,C)_berechnet:',time.time()-startTime, 's')  
print('K', K, '\nM', M, '\nC', C)  
  
'''  
ZEITSCHRITTVERFAHREN:  
Newmark-Beta-Verfahren  
'''  
U= timeIntegrationMethod.newmarkBetaMethod(U, V, A, K, M, C, w,
```

---

```
w_der, h_t, m, a_, nSF)

print('Laufzeit_nach_FEM_und_Zeitschrittverfahren:',
      time.time()-startTime, 's')

return U
```

## 10.4 integration.py

```
import numpy as np
import shapeFunctions
from settings_domain1D import *

'''

Legendre Polynome ueber 3-Term-REkursion berechnet
wird in Pderived benoetigt
Matrix T= [L0, L1, L2, ..., Ln]
'''

def LegendrePolynomials(T):
    n= T.shape[0]

    T[0][0]= 1 #erster Spaltenvektor (1)
    T[1][1]= 1 #zweiter Spaltenvektor (x)
    for j in range(1, n-1):# Spalte
        for i in range(0, n): #Zeile
            T[i][j+1]= (2*j+1)/(j+1)*T[i-1][j]- (j)/(j+1)*T[i][j-1]
    return T

'''

n-te Legendre Polynom abgeleitet und an der Stelle x ausgewertet
um die Gewichte der Gauss Legendre Quadraturformel zu bestimmen
'''

def Pderived(n, x):
    S= 0
    T= np.zeros(shape=(n+1, n+1))#+1 (0. bis n.-Legendre Polynom)
    LegendrePolynomials(T)
    for i in range(1, n+1):# wegen Ableitung 0->1 (shift)
        S+= T[i][n]* (i)* x**i
    return S

'''
```

```
GAUSS-LEGENDRE-QUADRATURE_RULE
I=[-1, 1]
(1) Stuetzstellen -> Nullstellen des n-ten Legendre-Polynoms.
3-Term-Rekursion-> Eigenwertproblem->Stuetzstellen
(2): aus Nullstellen Gewichte berechnen ueber Formel (benoetigt Pderived)
Ruekgabe: 2 Listen -> [Liste der Stuetzstellen, Liste der Gewichte]
'''

def GaussLegendreQuadratureRule(n): #order
    T = np.zeros(shape=(n, n))

    T[0][0] = 0
    T[0][1] = 1
    for i in range(1, n-1):
        #
        ai = ((i+1) - 1)/(2*(i+1) - 1)
        bi = ((i+1))/ (2*(i+1) - 1)
        #
        T[i][i - 1] = ai
        T[i][i] = 0
        T[i][i+1] = bi
        #
        an = (n - 1)/(2*n - 1)
    T[n - 1][n - 2] = an

    points = np.linalg.eigvals(T)
    points.sort()

    #--weights--
    weights = np.zeros(len(points))
    for i in range(0, len(points)):
        weights[i] = (2/(1 - points[i] ** 2)) / (Pderived(n, points[i]))**2
    return [points, weights]

'''

Transformation von X auf x des k-ten Elements
[-1, 1] auf [xk, xk+1]
'''
def transform_X_to_x(k, X, h_x):
    return k*h_x + (h_x)*(X+1)/2
```

```

"""
Einsfunktion
wird fuer die Berechnung der Eintraege der Steifigkeitsmatrix benoetigt
"""

def Eins(x):
    return 1


"""
Funktion
wird fuer die Berechnung der Eintraege der Massenmatrix benoetigt
"""

def mue_eps(x):
    return mue0*eps(x)


"""
Funktion
wird fuer die Berechnung der Eintraege der Daempfungsmatrix benoetigt
"""

def mue_sigma(x):
    return mue0*sigma(x)

#####
# Berechnung der lokalen Elementmatrix des k-ten Elements
# -Berechnung der k-ten Elementmatrix:
# Jede lokale Basisfunktion p_i mit jeder lokalen Basisfunktion p_j multipliziert
# und ueber das k-te Finite Element integriert.
#


def calculateLocalMatrix(element_k, derived, xi, w, A, f, h_x, nSF, SF):
    for i in range(0, np.shape(A)[0]):
        for j in range(0, np.shape(A)[0]):
            #Gauss-quad
            A[i][j]=quadrature(element_k, derived, i, j, f, xi, w, nSF, h_x, SF)
    return A


"""
Zusammenhang zwischen Eintraegen der lokalen Element-Matrizen und
Eintraegen der globalen Matrix.
k... k-tes finite Element
"""

```

```
'''  
def local_global_table(k, A_local, A_global, nSF):  
    for i in range(0, nSF+1):  
        for j in range(0, nSF+1):  
            A_global[nSF*k +i][nSF*k +j] += A_local[i][j]  
    return 1  
  
'''  
Gauss-Legendre Quadratur:  
Uebergabeparameter:  
-element_k...k-te finte Element  
-derived...Unterscheidung ob Basisfunktion od. abgeleitete Basisfunktion  
-(i, j)... Eintrag in der lokalen Matrix der berechnet wird  
    ->(Zugriff auf jeweilige Basisfunktionen)  
-(xi, w)... Listen an Stuetzstellen und Gewichten  
Ruekgabe: Naehlerungsloesung des Integrals des Eintrags(i, j)  
der k-ten lokalen Elementmatrix A_k  
'''  
def quadrature(element_k, derived, i, j, f, xi, w, nSF, h_x, SF):  
    Q= 0  
    for k in range(0, len(w)):  
        if(derived== 0):  
            if(SF==0):  
                Q+= w[k] * f(transform_X_to_x(element_k, xi[k], h_x))*  
                    h_x/2* shapeFunctions.Lagrange(nSF, i, xi[k], h_x)*  
                    shapeFunctions.Lagrange(nSF, j, xi[k], h_x)  
            else:  
                Q+= w[k] * f(transform_X_to_x(element_k, xi[k], h_x))*  
                    h_x/2* shapeFunctions.Legendre(nSF, i, xi[k], h_x)*  
                    shapeFunctions.Legendre(nSF, j, xi[k], h_x)  
        else:  
            if(SF==0):  
                Q+= w[k] * f(transform_X_to_x(element_k, xi[k], h_x))*  
                    h_x/2*shapeFunctions.Lagrange_der(nSF,i, xi[k],h_x)*  
                    shapeFunctions.Lagrange_der(nSF, j, xi[k], h_x)  
            else:  
                Q+= w[k] * f(transform_X_to_x(element_k, xi[k], h_x))*  
                    h_x/2*shapeFunctions.Legendre_der(nSF,i, xi[k],h_x)*  
                    shapeFunctions.Legendre_der(nSF, j, xi[k], h_x)  
    return Q
```

## 10.5 timeIntegrationMethod.py

```

import numpy as np
import integration

'''

Loesen eines tridiagonalen Gleichungssystems
(nur bei linearen Basisfunktionen verwendbar)
'''

def TDSolver(A, x, d):
    n= len(x)
    array_ci= np.zeros(n)
    array_di= np.zeros(n)

    array_ci[0]= A[0][1]/A[0][0] #c1/b1
    array_di[0]= d[0]/A[0][0]     #c1/b1
    for i in range(1, len(x)-1):
        array_ci[i]= A[i][i+1]/(A[i][i]- A[i][i-1]*array_ci[i-1])
    for i in range(1, len(x)):
        array_di[i]= (d[i]-A[i][i-1]*array_di[i-1])
                           /(A[i][i]-A[i][i-1]*array_ci[i-1])
    x[n-1]= array_di[n-1]
    for i in range(n-2, -1, -1):
        x[i]= array_di[i]-array_ci[i]*x[i+1]

    '''

NEWMARK-BETA-VERFAHREN:
Loesungsmatrix U, Geschwindigkeitsmatrix V, Beschleunigungsmatrix A
Systemmatrizen: Steifigkeitsmatrix K, Massenmatrix M, Daempfungsmatrix C
Loesung eines Zeitschrittes wird in einem Spaltenvektor der Matrix U,V,A
gespeichert.
'''

def newmarkBetaMethod(U, V, A, K, M, C, w, w_der, h_t, m, a_, nSF):
    gamma= 1/2
    beta= 1/4

    b = np.zeros(np.shape(K)[0])

    U[0][0] = w(0)      #Anfangszustand
    V[0][0] = w_der(0) #Anfangsgeschwindigkeit

    MatrixAi = M / beta / h_t ** 2 + gamma / beta / h_t * C + K

    for i in range(1, m):

```

```
b[0]= (np.sqrt(integration.mue_eps(a_))) *2*w_der(i*h_t)
VektorBi= b+ M.dot(U[:,i-1]/beta/h_t**2+
                     V[:,i-1]/beta/h_t+(1/2/beta-1)*A[:,i-1]) +
                     C.dot(gamma*U[:,i-1]/beta/h_t-
                     V[:,i-1]*(1-gamma/beta)-h_t*A[:,i-1]*(1-gamma/2/beta))
if (nSF > 1): #Ordnung der Formfunktionen
    U[:, i] = np.linalg.solve(MatrixAi, VektorBi)
else:
    TDSolver(MatrixAi, U[:, i], VektorBi)
A[:,i]= (U[:,i]-U[:,i-1])/beta/h_t**2-
         V[:,i-1]/beta/h_t-(1/2/beta-1)*A[:,i-1]
V[:,i]= gamma *(U[:, i]-U[:, i-1])/beta/h_t+
         V[:, i-1]*(1-gamma/beta) + h_t*A[:, i-1]*(1-gamma/2/beta)
return U
```

## 10.6 shapeFunctions.py

```
'''  
Lagrange Formfunktionen beliebiger Ordnung n  
[-1,1]def Lagrange(n, j, X, h_x):  
    product = 1  
    for k in range(0, n+1):  
        if (k != j):  
            product *= (X-(-1+k* 2/n)) / ((-1+j * 2/n) - (-1+k * 2/n))  
    return product  
  
'''  
abgeleitete Lagrange Formfunktionen beliebiger Ordnung n  
[-1,1]  
innere Ableitung (2/h_x), bei Integration benoetigtdef Lagrange_der(n, j, X, h_x):  
    sum = 0  
    for k in range(0, n+1):  
        product = 1  
        if (k != j):  
            for m in range(0, n + 1):  
                if (m != j and m != k):  
                    product *=(X-(-1+m * 2/n)) / ((-1+j*2/n) - (-1+m*2/n))  
    sum += product / ((-1 + j * 2 / n) - (-1 + k * 2 / n))
```

```

    return sum*(2/h_x)

'''  

Legendre Formfunktionen beliebiger Ordnung n  

[-1,1]
'''  

def Legendre(order, n, X, h_x):  

    l1 = (1 - X) / 2  

    l2 = (X + 1) / 2  

    if (n == 0):  

        return l1  

    if(n== order):  

        return l2  

    li_minus_1= -1  

    li= X  

    for i in range(2, n + 2):  

        l = li  

        li = (2 * i - 3) / (i) * X * li - (i - 3) / (i) * li_minus_1  

        li_minus_1 = l  

    return li

'''  

abgeleitete Legendre Formfunktionen beliebiger Ordnung n  

[-1,1]  

innere Ableitung (2/h_x), bei Integration benoetigt
'''  

def Legendre_der(order, n, X, h_x):  

    l1 = (-1) /2  

    l2 = (1) /2  

    if (n == 0):  

        return l1 *(2/h_x)  

    if (n == order):  

        return l2 *(2/h_x)  

    # n>1  

    ln_minus1=1  

    ln = X  

    if(n == 1):

```

```
    return ln * (2/h_x)

for i in range(2, n+1):
    l = ln
    ln = (2*i-1) / (i) * X * ln - (i - 1) / i * ln_minus1
    ln_minus1 = l
return ln * (2/h_x)
```

## 10.7 plotResult.py

```
from matplotlib import animation
from matplotlib import pyplot as plt
from settings_domain1D import *
import numpy as np
import shapeFunctions

resolution_x= 200
resolution_m= 1000

'''

Da die Spaltenvektoren der Naeherungsloesung U die Koeffizienten enthalten muss um den Loesungswert bei x auszuwerten die relevanten Basisfunktionen bei x ausgewertet und mit den jeweiligen Koeffizienten gewichtet werden

'''

def getSolutionValueFromU(U, timestep, x, nSF, h_x, nFE, SF):
    coeffs= np.zeros(nSF+1)
    k= int(x/h_x)
    if(k>=nFE):
        k= k-1

    for i in range(0, nSF+1):
        coeffs[i]= U[k*nSF+i][timestep]
    X= (x-k*h_x)*2/h_x -1

    sum= 0
    if(SF==0):
        for i in range(0, nSF+1):
            sum+= coeffs[i]*shapeFunctions.Lagrange(nSF, i, X,
h_x)
    else:
        for i in range(0, nSF+1):
```

```

    sum+= coeffs[i]*shapeFunctions.Legendre(nSF, i, X,
h_x)
    return sum

"""

Animation um die Loesung U darzustellen
Spaltenvektoren von U enthalten Koeffizienten der
Basisfunktionen
Python Animations Bsp.:
(vgl. https://jakevdp.github.io/blog/2012/08/18/matplotlib-animation-tutorial/)
"""

def animation2D(U, m, nSF, h_x, nFE, SF):
    if m>=resolution_m:
        m_new= resolution_m
    else:
        m_new= m
    U_nodes = np.zeros(shape=(resolution_x+1, m_new))
    X = np.zeros(shape=(resolution_x+1, m_new))

    #Problemgebiet [a,b]
    for i in range(0, resolution_x+1):
        for j in range(0, m_new):
            X[i][j]= a_+ (b_-a_)/resolution_x*i

    for i in range(0, resolution_x+1):
        for j in range(0, m_new):
            U_nodes[i][j]= getSolutionValueFromU(U, int(j*m/
m_new), X[i][j], nSF, h_x, nFE, SF)

    fig = plt.figure()
    ax = plt.axes(xlim=(a_- (b_-a_)*0.05, b_+ (b_-a_)*0.05),
    ylim=(-2,2))
    data, = ax.plot([], [], lw=2)
    text_for_timestep = ax.text(0.045, 0.90, '', transform=ax.
transAxes,
    fontsize= 15)

    def initialisation():
        data.set_data([], [])
        text_for_timestep.set_text('')


```

```
    return data, text_for_timestep

def anima(j):
    data.set_data(X[:, j], U_nodes[:, j])
    text_for_timestep.set_text('$t/t_0=$' + str((round(j*1/m
*10))/10))
    return data, text_for_timestep

anm= animation.FuncAnimation(fig, anima, init_func=
initialisation,
                             frames=m_new, interval= 30, blit=True)

ax.set_xlabel('$x[m]$', fontsize= 15)
ax.set_ylabel('$u(x,t)[V/m]$', fontsize=15)
ax.tick_params(axis="x", labelsize=14)
ax.tick_params(axis="y", labelsize=14)

plt.grid(True)
plt.show()
```

# Literaturverzeichnis

- [Arn07] Anton Arnold. Open boundary conditions for wave propagation problems on unbounded domains, <https://www.asc.tuwien.ac.at/~arnold/pdf/graz/graz.pdf>, 9 2007.
- [AS72] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions*. National Bureau of Standards, 1972.
- [Dat10] Biswa Nath Datta. *Numerical Linear Algebra and Applications, Second Edition*. Society for Industrial and Applied Mathematics, USA, 2nd edition, 2010.
- [GW69] Gene H. Golub and John H. Welsch. Calculation of Gauss quadrature rules, <https://doi.org/10.1090/S0025-5718-69-99647-1>. *Mathematics of Computation*, 23:221–230, 1969.
- [HMRH13] S. Habib Mazharimousavi, Ashkan Roozbeh, and M. Halilsoy. Electromagnetic wave propagation through inhomogeneous material layers. *Journal of Electromagnetic Waves and Applications*, 27(16):2065–2074, Sep 2013.
- [Hol20] Karl Hollaus. Einführung in partielle Differentialgleichungen und Methode der finiten Elemente, Vorlesungsmanuskript, Juli 2020.
- [JL13] Michael Jung and Ulrich Langer. *Methode der finiten Elemente für Ingenieure*. Springer Vieweg, Wiesbaden, 2013.
- [LP19] George Lindfield and John Penny. Chapter 5 - solution of differential equations. In George Lindfield and John Penny, editors, *Numerical Methods (Fourth Edition)*, pages 239–299. Academic Press, fourth edition, 2019.
- [Mec20] Christoph Mecklenbräuker. Wellenausbreitung, Vorlesungsmanuskript, 2020.
- [Pre10] A. Prechtl. Vorlesung über Elektrodynamik, Vorlesungsmanuskript, 2010.
- [Rei12] Georg A. Reider. *Photonik, Eine Einführung in die Grundlagen*. Springer-Verlag Wien, 2012.
- [Whi17] Jonathan Whiteley. *Finite Element Methods, A Practical Guide*. Springer Nature, Cham, Schweiz, 2017.