Freie Universität Berlin

Fachbereich Mathematik und Informatik
Institut für Mathematik

Bachelorarbeit

eingereicht zum akademischen Abschluss

Bachelor of Science

Finite-Differenzen-Verfahren

für konvektionsdominante Zwei-Punkt-Randwertprobleme

Student:	Louise Richter
Matrikelnummer:	5576124
Erstprüfer:	Prof. Dr. Volker John
Zweitprüfer:	Dr. Alfonso Caiazzo
Abgabedatum:	22.12.2025

Freie Universität Berlin

Fachbereich Mathematik, Informatik und Physik

SELBSTSTÄNDIGKEITSERKLÄRUNG

Name:	(BITTE nur Block- oder Maschinenschrift verwenden.)
Vorname(n):	
Studiengang:	
Matr. Nr.:	

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende _____ selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht.

Datum: _____ Unterschrift: _____

Im Sinne guter wissenschaftlicher Praxis erkläre ich, dass KI-gestützte Werkzeuge (z. B. ChatGPT von OpenAI) ausschließlich zur sprachlichen Überarbeitung und Formatierung eingesetzt wurden.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Differentialgleichungen spielen bei der Modellierung vieler Prozesse in der Physik eine wichtige Rolle. Man betrachte beispielsweise einen Fluss mit starker, gleichmäßiger Strömung. An einer bestimmten Stelle fließt ein flüssiger Schadstoff in das Wasser. Wir stellen die Frage, welche Form die entstehende Verschmutzungsfahne auf der Wasseroberfläche annimmt.
Das Verhalten wird durch zwei Prozesse bestimmt: Der Stoff diffundiert langsam durch das Wasser; der dominante Prozess ist jedoch die Strömung des Flusses, welche den Stoff wesentlich schneller flussabwärts transportiert (konvektiert). Durch Konvektion allein würde sich der Stoff entlang einer eindimensionalen Kurve an der Oberfläche bewegen. Die Diffusion führt jedoch zu einer allmählichen Ausbreitung um diese Kurve herum, sodass schließlich eine langgezogene, schmale, keilförmige Verteilung entsteht.
Mathematisch führt dies auf ein Konvektions-Diffusions-Problem, bei dem Diffusion und Konvektion gleichzeitig auftreten, die Konvektion aber wie im Beispiel dominiert. Man kann diesen Prozess mithilfe eines linearen Zwei-Punkt-Randwertproblems der Form

$$-\varepsilon u''(x) + b(x)u'(x) + \sigma(x)u(x) = f(x), \qquad 0 < x < 1,$$

mit Randbedingungen

$$u(0) = u(1) = 0,$$

modellieren. Dabei ist $\varepsilon > 0$ ein Parameter und b, σ und f sind gegebene Funktionen .
Der Term u'' modelliert die Diffusion mit kleinem Diffusionskoeffizienten ε. Der Term u' steht für die Konvektion, während σu als Reaktionsterm die lokale Zu- oder Abnahme von u ohne Transport beschreibt. Der Term f wirkt als Quellenterm und modelliert eine äußere Zuführung oder Entnahme, die nicht aus dem System selbst entsteht.
Dominiert die Konvektion, entstehen in der Lösung oft sogenannte Grenzschichten. Das sind Bereiche, auf denen die Lösung auf einem kleinen Intervall starke Änderungen zeigt. Für diesen Fall wollen wir die numerische Approximation genauer untersuchen.
In dieser Arbeit betrachten wir ausschließlich eindimensionale lineare Zwei-Punkt-Randwertprobleme. Diese treten beispielsweise in der Wärmeleitung, in der Strömungsmechanik, in chemischen Reaktionen, in elektrischen Netzwerken oder auch in Transportprozessen auf.
Ziel dieser Arbeit ist es, das analytische Verhalten solcher singulär gestörten Randwertprobleme zu untersuchen. Dabei werden numerische Verfahren betrachtet, um auch im konvektionsdominanten Fall stabile und genaue Approximationen zu erhalten.
Dazu wird in Kapitel 2 zunächst das stetige Problem eingeführt und das Verhalten der Lösung analysiert. Kapitel 3 behandelt Finite-Differenzen-Methoden und die damit ver-

bundene Fehleranalyse. In Kapitel 4 wird explizit der konvektionsdominante Fall untersucht, in dem Grenzschichten besonders stark ausgeprägt sind. In Kapitel 5 werden stabile numerische Verfahren wie M-Matrizen, das Upwind-Verfahren und gleichmäßig konvergente Methoden vorgestellt. Ziel dieser Arbeit ist es, sowohl theoretische Einsichten als auch praktische numerische Methoden für singulär gestörte Konvektions-Diffusions-Probleme zu vermitteln.

Kapitel 2

Stetiges Problem

2.1 Allgemeine lineare Zwei-Punkt-Randwertprobleme

Wir folgen der Definition aus [5, p. 3].
Seien $x \in (\alpha, \beta)$ sowie $b, \sigma, f \in C([\alpha, \beta])$ und $\varepsilon \in \mathbb{R}$ mit $\varepsilon > 0$.

Wir betrachten das allgemeine lineare Zwei-Punkt-Randwertproblem der Form

$$-\varepsilon u''(x) + b(x)u'(x) + \sigma u(x) = f(x), \tag{2.1}$$

versehen mit verallgemeinerten linearen Randbedingungen

$$\lambda_\alpha u(\alpha) - \mu_\alpha u'(\alpha) = \nu_\alpha, \tag{2.2}$$

$$\lambda_\beta u(\beta) - \mu_\beta u'(\beta) = \nu_\beta. \tag{2.3}$$

Bemerkung 2.1.1 (Randbedingungen) [4, p. 5–6]
Für $\nu_\alpha, \nu_\beta \in \mathbb{R}$ und $\lambda_\alpha, \lambda_\beta \in \mathbb{R} \setminus \{0\}$ unterscheidet man zwischen folgenden Randbedingungen:

1. **Dirichlet-Randbedingungen (1. Art):**

$$u(\alpha) = \nu_\alpha, \qquad u(\beta) = \nu_\beta.$$

2. **Neumann-Randbedingungen (2. Art):**

$$u'(\alpha) = \nu_\alpha, \qquad u'(\beta) = \nu_\beta.$$

3. **Robin-Randbedingungen (3. Art):**

$$\lambda_\alpha u(\alpha) + u'(\alpha) = \nu_\alpha, \qquad \lambda_\beta u(\beta) + u'(\beta) = \nu_\beta.$$

Bemerkung 2.1.2 (Homogene Randbedingungen) [2, p. 1]
Eine Randbedingung heißt *homogen*, wenn sie auf dem Rand eines Gebietes $\Omega \subseteq \mathbb{R}^n$ den Wert 0 annimmt. Sonst heißt sie *inhomogen*.

Bemerkung 2.1.3 (Normierung) [5, p. 4-5]
Zur Vereinfachung transformieren wir das Randwertproblem in eine Standardform:

1. Ohne Beschränkung der Allgemeinheit sei $x \in [0,1]$. Das wird erreicht durch die lineare Transformation
$$T(x) = \frac{x-\alpha}{\beta-\alpha}.$$

2. Seien die Randbedingungen o.B.d.A homogen, also $\nu_\alpha = 0$ und $\nu_\beta = 0$. Dies erhält man durch die Substitution $u \mapsto u - g$, wobei g eine glatte Funktion ist, die die ursprünglichen Randbedingungen erfüllt.

Definition 2.1.4 (Standardproblem) [5, p. 5]
Seien $x \in (0,1)$ und $b(x), \sigma(x), f(x) \in C([0,1])$ sowie $\varepsilon \in \mathbb{R}$ mit $\varepsilon > 0$. Dann lautet das Standardproblem:

$$Lu := -\varepsilon u'' + bu' + \sigma u = f \quad \text{in} \quad \Omega = (0,1) \tag{2.4}$$

mit den Randbedingungen

$$u(0) = u(1) = 0. \tag{2.5}$$

Bemerkung 2.1.5 (Zum Operatorbegriff) [4, p. 7]
In (2.4) bezeichnet L einen Differentialoperator. Unter einem Operator versteht man im Allgemeinen eine Abbildung zwischen zwei (Funktionen-)Räumen.
Ein linearer Operator ist eine lineare Abbildung A auf einem Vektorraum X mit der Eigenschaft

$$A(\lambda u + \mu v) = \lambda Au + \mu Av,$$

wobei λ, μ beliebige Skalare und $u, v \in X$ beliebige Elemente sind.
Ein *Differentialoperator* erzeugt bei Anwendung Ableitungen. Zu seiner vollständigen Definition gehört stets die Angabe des Definitionsbereiches.

Definition 2.1.6 (Reduziertes Problem) [4, p. 8]
Um das reduzierte Problem zu erhalten, setzt man $\varepsilon = 0$. Damit ergibt sich die Differentialgleichung

$$L_0 u_0 := b(x)\, u_0'(x) + \sigma(x)\, u_0(x) = f(x), \qquad x \in (0,1).$$

Die Randbedingung wird an der Einströmkante gesetzt:

- Ist $b(x) > 0$ für alle $x \in [0,1]$, so wird die Randbedingung wird bei $x = 0$ gesetzt:
$$u_0(0) = 0.$$

- Ist $b(x) < 0$ für alle $x \in [0,1]$, so wird die Randbedingung bei $x = 1$ gesetzt:
$$u_0(1) = 0.$$

Die Lösung u_0 dieses Problems wird reduzierte Lösung genannt.

2.2 Analytisches Verhalten der Lösung

Definition 2.2.1 [3, p. 13]
Die folgenden Funktionenräume werden benötigt:

- Der Raum $C(a,b)$ umfasst alle Funktionen

$$u : (a,b) \to \mathbb{R},$$

die auf dem offenen Intervall (a,b) stetig sind.

- Der Raum $C[a,b]$ umfasst alle Funktionen

$$u : [a,b] \to \mathbb{R},$$

die auf dem abgeschlossenen Intervall $[a,b]$ stetig sind.

- Der Raum $C^2(a,b)$ umfasst alle Funktionen

$$u : (a,b) \to \mathbb{R},$$

die zweimal stetig differenzierbar sind, d. h.

$$u,\ u',\ u'' \text{ sind stetig auf } (a,b).$$

Bemerkung 2.2.2 (Standardproblem) [4, p. 8]
Die Lösbarkeit des Problems (2.4)-(2.5) ist unabhängig von $\varepsilon > 0$. Durch Division durch ε und geeignete Umbenennung der Koeffizienten erhält man

$$Lu := -u''(x) + b(x)u'(x) + \sigma(x)u(x) = f(x), \qquad x \in (0,1), \tag{2.6}$$

mit den Randbedingungen

$$u(0) = u(1) = 0. \tag{2.7}$$

Definition 2.2.3 (Klassische Lösung eines Randwertproblems) [3, p. 13]
Wir betrachten ein Randwertproblem mit Dirichlet-Randbedingungen für eine Differentialgleichung zweiter Ordnung auf dem Intervall (a,b). Eine *klassische Lösung* ist eine Funktion

$$u \in C^2(a,b) \cap C[a,b],$$

die sowohl die Differentialgleichung als auch die Randbedingungen erfüllt.

Definition 2.2.4 (Wronski-Determinante) [3, p. 14]
Für zwei Funktionen u_1, u_2 auf einem Intervall I ist die Wronski-Determinante definiert durch

$$W(x) = \begin{vmatrix} u_1(x) & u_2(x) \\ u_1'(x) & u_2'(x) \end{vmatrix}, \quad x \in I.$$

Definition 2.2.5 (Lineare Unabhängigkeit) [3, p. 14]
Zwei Funktionen $u_1(x)$ und $u_2(x)$ heißen *linear unabhängig*, wenn aus

$$c_1 u_1(x) + c_2 u_2(x) = 0 \quad \text{für alle } x \in (a,b) \tag{2.8}$$

folgt, dass

$$c_1 = c_2 = 0.$$

Sind sie nicht linear unabhängig, so heißen sie *linear abhängig*.

Bemerkung 2.2.6 [3, p. 14]
Seien u_1, u_2 auf (a, b) linear abhängig und stetig differenzierbar. Dann folgt aus (2.8), dass

$$c_1 u_1'(x) + c_2 u_2'(x) = 0, \quad x \in (a, b). \tag{2.9}$$

Damit sind auch u_1' und u_2' linear abhängig. Folglich besitzt das Gleichungssystem aus (2.8) und (2.9) eine nichttriviale Lösung, und die Wronski-Determinante $W(x)$ ist identisch null.

Satz 2.2.7 (Superpositionsprinzip) [3, p. 16]
Für die homogene lineare Differentialgleichung

$$-u''(x) + b(x)u'(x) + \sigma(x)u(x) = 0, \qquad x \in (0, 1),$$

mit $b, \sigma \in C([0, 1])$, existieren zwei linear unabhängige Lösungen in $C^2([0, 1])$. Jede klassische Lösung lässt sich als Linearkombination dieser beiden Funktionen darstellen.

Satz 2.2.8 (Klassische Lösung der inhomogenen, linearen Differentialgleichung) [3, p. 17]
Für die inhomogene lineare Differentialgleichung

$$-u''(x) + b(x)u'(x) + \sigma(x)u(x) = f(x), \qquad x \in (0, 1),$$

mit $b, \sigma, f \in C([0, 1])$, existiert eine klassische partikuläre Lösung u_p. Jede klassische Lösung hat die Darstellung

$$u(x) = c_1 u_1(x) + c_2 u_2(x) + u_p(x), \qquad c_1, c_2 \in \mathbb{R},$$

wobei u_1, u_2 ein Fundamentalsystem der zugehörigen homogenen Gleichung bilden, also eine Basis des Lösungsraums der zugehörigen homogenen linearen Differentialgleichung.

Satz 2.2.9 (Existenz und Eindeutigkeit der Lösung des Modellproblems mit homogener rechter Seite) [4, p. 12]
Für das Randwertproblem (2.6), (2.7) mit

$$b \in C^1([0, 1]), \qquad \sigma \in C([0, 1]), \qquad f(x) \equiv 0,$$

gilt: Ist für alle $x \in (0, 1)$

$$\tilde{\sigma}(x) := \frac{1}{4} b^2(x) - \frac{1}{2} b'(x) + \sigma(x) \geq 0,$$

dann besitzt das Problem (2.6), (2.7) nur die triviale Lösung, also

$$u(x) \equiv 0.$$

Bemerkung 2.2.10 (Konstante Koeffizienten) [4, p. 13]
Für konstante Koeffizienten vereinfacht sich die Bedingung $\tilde{\sigma}(x) \geq 0$ zu

$$D := \frac{b^2}{4} + \sigma \geq 0.$$

Definition 2.2.11 (Invers-monotoner Operator) [9, p. 10]
Sei $M : C^2(0,1) \to C(0,1)$ ein Differentialoperator und $w \in C^2(0,1) \cap C[0,1]$. Der Operator M heißt invers-monoton, wenn aus

$$Mw(x) \geq 0 \quad \text{für alle } x \in (0,1), \qquad w(0) \geq 0, \quad w(1) \geq 0$$

folgt, dass

$$w(x) \geq 0 \quad \text{für alle } x \in [0,1].$$

Das bedeutet: Ein Operator ist invers-monoton, wenn aus der Nichtnegativität des Operatorausdrucks und der Randwerte folgt, dass die Funktion auf dem gesamten Intervall nichtnegativ ist.

Bemerkung 2.2.12 [9, p. 10]
Für $\sigma(x) \geq 0$ für alle $x \in [0,1]$ ist der Operator L aus (2.4), (2.5) invers-monoton.

Definition 2.2.13 (Maximumprinzip) [9, p. 10]
Ein Differentialoperator $M : C^2(0,1) \to C(0,1)$ erfüllt ein *Maximumprinzip*, wenn für jede Funktion $u \in C^2(0,1) \cap C[0,1]$ aus

$$Mu(x) = 0 \quad \text{für alle } x \in (0,1)$$

folgt, dass

$$\min\{u(0), u(1), 0\} \;\leq\; u(x) \;\leq\; \max\{u(0), u(1), 0\} \quad \text{für alle } x \in [0,1].$$

Bemerkung 2.2.14 [9, p. 10]
Ist der Operator L invers-monoton, so erfüllt er auch das Maximumprinzip und das folgende Vergleichsprinzip.

Definition 2.2.15 (Vergleichsprinzip) [9, p. 10]
Seien $v, w \in C^2(0,1) \cap C[0,1]$. Es gelte

$$Lw(x) \geq Lv(x) \quad \text{für alle } x \in (0,1) \qquad \text{und} \qquad w(0) \geq v(0), \quad w(1) \geq v(1).$$

Dann gilt

$$w(x) \geq v(x) \quad \text{für alle } x \in [0,1].$$

Wir nennen w eine Barrierefunktion für v.

2.3 Lineare Probleme zweiter Ordnung ohne Wendepunkte

Dieser Abschnitt folgt [9, p. 11-16].
Wir betrachten das Randwertproblem

$$Lu := -\varepsilon u''(x) + b(x)u' + \sigma(x)u = f(x), \qquad x \in (0,1), \quad \sigma(x) \geq 0 \tag{2.10}$$

mit den Randbedingungen

$$u(0) = u(1) = 0. \tag{2.11}$$

Für hinreichend glatte Funktionen b, σ, f existiert eine eindeutige klassische Lösung. Unser Ziel ist es, das Verhalten dieser Lösung für kleine $\varepsilon > 0$ zu untersuchen.

2.3.1 Asymptotische Entwicklungen

Wir wollen nun untersuchen, wie wir u aus (2.10), (2.11) durch eine einfache Funktion approximieren können. Die Methode der *matched asymptotic expansions* liefert eine systematische Approximation von u.

Definition 2.3.1 (Asymptotische Entwicklung)
Eine Funktion u_{as} heißt asymptotische Entwicklung der Ordnung m von der Lösung u (in der Maximumsnorm), wenn eine Konstante $C > 0$ existiert, sodass

$$|u(x) - u_{\text{as}}(x)| \leq C\varepsilon^{m+1}$$

für alle $x \in [0, 1]$ und alle hinreichend kleinen $\varepsilon > 0$.

Dabei ist C unabhängig von ε und b, σ, f sind ausreichend glatt auf $[0, 1]$.

Globale Erweiterung

Wir versuchen zunächst, eine globale Erweiterung u_g zu finden. Diese Funktion wird eine gute Näherung von u außerhalb der Grenzschicht(en) sein, d. h. auf fast dem gesamten Intervall $[0, 1]$. Wir setzen

$$u_g(x) = \sum_{\nu=0}^{m} \varepsilon^\nu u_\nu(x),$$

wobei $u_\nu(x)$ noch bestimmt werden müssen.

Definition 2.3.2 (Reduzierter Operator)
Durch formales Setzen von $\varepsilon = 0$ in L erhalten wir den Operator

$$L_0 v := bv' + \sigma v.$$

Als nächstes setzen wir u_g in die Gleichung (2.10), (2.11) ein. Durch Gleichsetzen der Koeffizienten gleicher Potenzen von ε erhalten wir die folgenden Gleichungen:

$$L_0 u_0 = f, \tag{2.12}$$
$$L_0 u_\nu = u''_{\nu-1}, \qquad \nu = 1, \ldots, m. \tag{2.13}$$

Definition 2.3.3 (Wendepunkt)
Ein Punkt $x_0 \in [0, 1]$ heißt Wendepunkt, wenn

$$b(x_0) = 0.$$

Bemerkung 2.3.4
Hat b einen Wendepunkt, entstehen Schwierigkeiten bei der Definition der Koeffizienten u_ν . Wir betrachten hier also nur den Fall

$$b(x) \neq 0 \qquad \text{für alle } x \in [0, 1].$$

Wahl der Randbedingung für das reduzierte Problem
Eine der Randbedingungen aus (2.11) sollte verwendet werden, um u_0 zu berechnen. Es stellt sich die Frage, welche Randbedingung wegzulassen ist.

Theorem 2.3.5 (Cancellation Law)

- Ist $b > 0$ auf $[0, 1]$, so liegt die Grenzschicht bei $x = 1$ und man lässt die Randbedingung bei $x = 1$ unberücksichtigt.
- Ist $b < 0$, so liegt die Grenzschicht bei $x = 0$ und man lässt die Randbedingung bei $x = 0$ unberücksichtigt.

Da die Transformation $x \mapsto 1 - x$ den Fall $b < 0$ in $b > 0$ überführt, ist es ausreichend, den Fall $b > 0$ genauer zu betrachten.
Für $b > 0$ gilt

$$L_0 u_0 = f, \qquad u_0(0) = 0, \tag{2.14}$$
$$L_0 u_\nu = u''_{\nu-1}, \qquad u_\nu(0) = 0, \qquad \nu = 1, ...m. \tag{2.15}$$

Wir nennen (2.14) - (2.15) das reduzierte Problem mit der reduzierten Lösung u_0.

Lokale (innere) Korrektur
Das Ziel der asymptotischen Entwicklung ist, eine Näherung von u zu finden, die für alle $x \in [0, 1]$ gilt. Die Näherung u_g erfüllt die Randbedingung an der Stelle $x = 1$ nicht. Deshalb führen wir eine lokale Korrektur von u_g in der Nähe von $x = 1$ ein.
Wir setzen

$$w := u - u_g.$$

Dann erfüllt w

$$Lw = \varepsilon^{m+1} u''_m,$$
$$w(0) = 0, \qquad w(1) = -\sum_{\nu=0}^{m} \varepsilon^\nu u_\nu(1).$$

Streckung der unabhängigen Variablen
Wir können L schreiben als $L = \varepsilon L_1 + L_0$. Da die Grenzschicht bei $x = 1$ liegt, strecken wir die Skala dort in x-Richtung durch die Variable

$$\xi = \frac{1 - x}{\delta},$$

wobei δ klein und noch unbestimmt ist. Wir wählen die Grenzschichtskalierung δ so, dass L_0 und εL_1 nach der Transformation von x nach ξ formal von derselben Ordnung bezüglich ε sind. Wir setzen

$$\varepsilon\, \delta^{-2} \;\approx\; \delta^{-1}.$$

Das führt zur Wahl

$$\delta = \varepsilon.$$

Wir entwickeln die Koeffizientenfunktionen b und σ bzgl. ξ mittels Taylorreihen um $x = 1$:

$$b(1-\varepsilon\xi) = \sum_{\nu=0}^{\infty} b_\nu \varepsilon^\nu \xi^\nu, \qquad b_0 = b(1),$$

$$\sigma(1-\varepsilon\xi) = \sum_{\nu=0}^{\infty} \sigma_\nu \varepsilon^\nu \xi^\nu, \qquad \sigma_0 = \sigma(1).$$

Damit lässt sich L für jede hinreichend differenzierbare Funktion g in der Variablen ξ darstellen als

$$\varepsilon L_1 g + L_0 g = \frac{1}{\varepsilon} \sum_{\nu=0}^{\infty} \varepsilon^\nu L_\nu^* g,$$

wobei

$$L_0^* := -\frac{d^2}{d\xi^2} \; - \; b_0 \frac{d}{d\xi}, \qquad L_1^* := -\, b_1 \xi \frac{d}{d\xi} \; + \; c_0,$$

usw.
Als lokale Korrektur verwenden wir

$$v_{\mathrm{loc}}(\xi) = \sum_{\mu=0}^{m+1} \varepsilon^\mu v_\mu(\xi).$$

Damit v_{loc} die Differenz

$$w = u - u_g$$

approximiert, müssen die Terme v_μ die Bedingungen

$$L_0^* v_0 = 0, \tag{2.16}$$

$$L_0^* v_\mu = -\sum_{\kappa=1}^{\mu} L_\kappa^* \, v_{\mu-\kappa}, \qquad \mu = 1, \ldots, m+1 \tag{2.17}$$

erfüllen.
Um die richtige Randbedingung bei $x = 1$ zu erhalten, wählen wir

$$v_\kappa(0) = -u_\kappa(1), \qquad \kappa = 0, 1, \ldots, m.$$

Da die Gleichungen (2.16) und (2.17) zweiter Ordnung sind, brauchen wir eine weitere Bedingung. Wir fordern

$$\lim_{\xi \to \infty} v_\mu(\xi) = 0.$$

Mit diesen beiden Randbedingungen besitzt das Problem (2.16), (2.17) eine eindeutige Lösung. Wir erhalten z.B. für die Korrektur erster Ordnung

$$v_0(\xi) = -u_0(1)\, e^{-\,b(1)\,\xi}.$$

Bemerkung 2.3.6
Grenzschichten werden entsprechend der zugehörigen Grenzschichtgleichungen klassifiziert. Die einfachsten Schichten sind exponentielle Grenzschichten (auch gewöhnliche Grenzschichten genannt), bei denen die Lösungen der Grenzschichtgleichungen abfallende Exponentialfunktionen sind. Die Lösung von (2.10), (2.11) hat normalerweise eine Grenzschicht dieses Typs bei $x = 1$, wenn $b > 0$ auf $[0, 1]$.

Theorem 2.3.7
Sind die Koeffizienten sowie die rechte Seite des Randwertproblems (2.10), (2.11) hinreichend glatt und gilt zudem $b(x) > \beta > 0$ auf $[0,1]$, so hat die Lösung u eine asymptotische Entwicklung der Form

$$u_{\text{as}}(x) = \sum_{\nu=0}^{m} \varepsilon^{\nu} u_{\nu}(x) \; + \; \sum_{\mu=0}^{m} \varepsilon^{\mu} v_{\mu}\left(\frac{1-x}{\varepsilon}\right),$$

sodass für jede hinreichend kleine Konstante $\varepsilon_0 > 0$ gilt:

$$|u(x) - u_{\text{as}}(x)| \; \leq \; C\,\varepsilon^{\,m+1} \qquad \text{für alle } x \in [0,1] \text{ und } \varepsilon \leq \varepsilon_0.$$

Dabei ist C unabhängig von x und ε.

Beweis. Der Beweis folgt [9, p. 15]. □

Beispiel 2.3.8
Wir wollen nun das Verfahren der asymptotischen Entwicklung an einem Beispiel näher betrachten. Wir betrachten das Randwertproblem

$$-\varepsilon u''(x) + u'(x) = 1, \qquad x \in (0,1), \tag{2.18}$$

mit den Randbedingungen

$$u(0) = u(1) = 0. \tag{2.19}$$

Nach Theorem 2.3.5 liegt eine Grenzschicht bei $x = 1$ vor, da $b(x) = 1 > 0$ gilt.

Reduziertes Problem

Wir setzen in (2.18) $\varepsilon = 0$ und erhalten somit

$$L_0 u_0 = u_0' = 1.$$

Mit der nach Theorem 2.3.5 gewählten Randbedingung $u_0(0) = 0$ folgt

$$u_0(x) = x. \tag{2.20}$$

Globale Erweiterung

Wir setzen

$$u_g(x) = u_0(x) + \varepsilon u_1(x) + \cdots .$$

Die Gleichung für u_1 ist nach (2.15)

$$L_0 u_1 = u_0'' = 0, \qquad u_1(0) = 0.$$

Damit folgt

$$u_1(x) = 0. \tag{2.21}$$

Bis zur Ordnung $O(\varepsilon)$ gilt daher

$$u_g(x) = x. \tag{2.22}$$

Lokale Grenzschicht bei $x = 1$

Da $b(1) = 1 > 0$ gilt, liegt eine Grenzschicht bei $x = 1$. Wir führen die gestreckte Variable

$$\xi = \frac{1 - x}{\varepsilon}$$

ein. Die lokale Korrektur schreiben wir in der Form

$$v_{\mathrm{loc}}(\xi) = v_0(\xi) + \varepsilon v_1(\xi) + \cdots .$$

Die Gleichung für v_0 ergibt sich aus (2.16):

$$-v_0''(\xi) - v_0'(\xi) = 0 \tag{2.23}$$

mit den Randbedingungen

$$v_0(0) = -u_0(1) = -1, \qquad \lim_{\xi \to \infty} v_0(\xi) = 0.$$

Gleichung (2.23) hat die Lösung

$$v_0(\xi) = -e^{-\xi}. \tag{2.24}$$

Asymptotische Gesamtlösung

Durch Addition der globalen und lokalen Anteile erhält man die asymptotische Lösung

$$u_{\mathrm{as}}(x) = u_g(x) + v_0\left(\frac{1 - x}{\varepsilon}\right) = x - e^{-(1-x)/\varepsilon}.$$

Vergleich mit der exakten Lösung

Das Randwertproblem (2.18)–(2.19) besitzt die exakte Lösung

$$u(x) = x - \frac{1 - e^{-x/\varepsilon}}{1 - e^{-1/\varepsilon}}. \tag{2.25}$$

Für $\varepsilon \to 0$ gilt

$$\frac{1 - e^{-x/\varepsilon}}{1 - e^{-1/\varepsilon}} = e^{-(1-x)/\varepsilon} + O\big(e^{-1/\varepsilon}\big),$$

und somit folgt

$$u(x) = x - e^{-(1-x)/\varepsilon} + O\big(e^{-1/\varepsilon}\big).$$

Damit stimmt die asymptotische Lösung mit der exakten Lösung bis auf einen exponentiell kleinen Fehler überein.
Wir betrachten die exakte und asymptotische Lösung für verschiedene Werte von ε. Der Fehler ist kaum sichtbar.

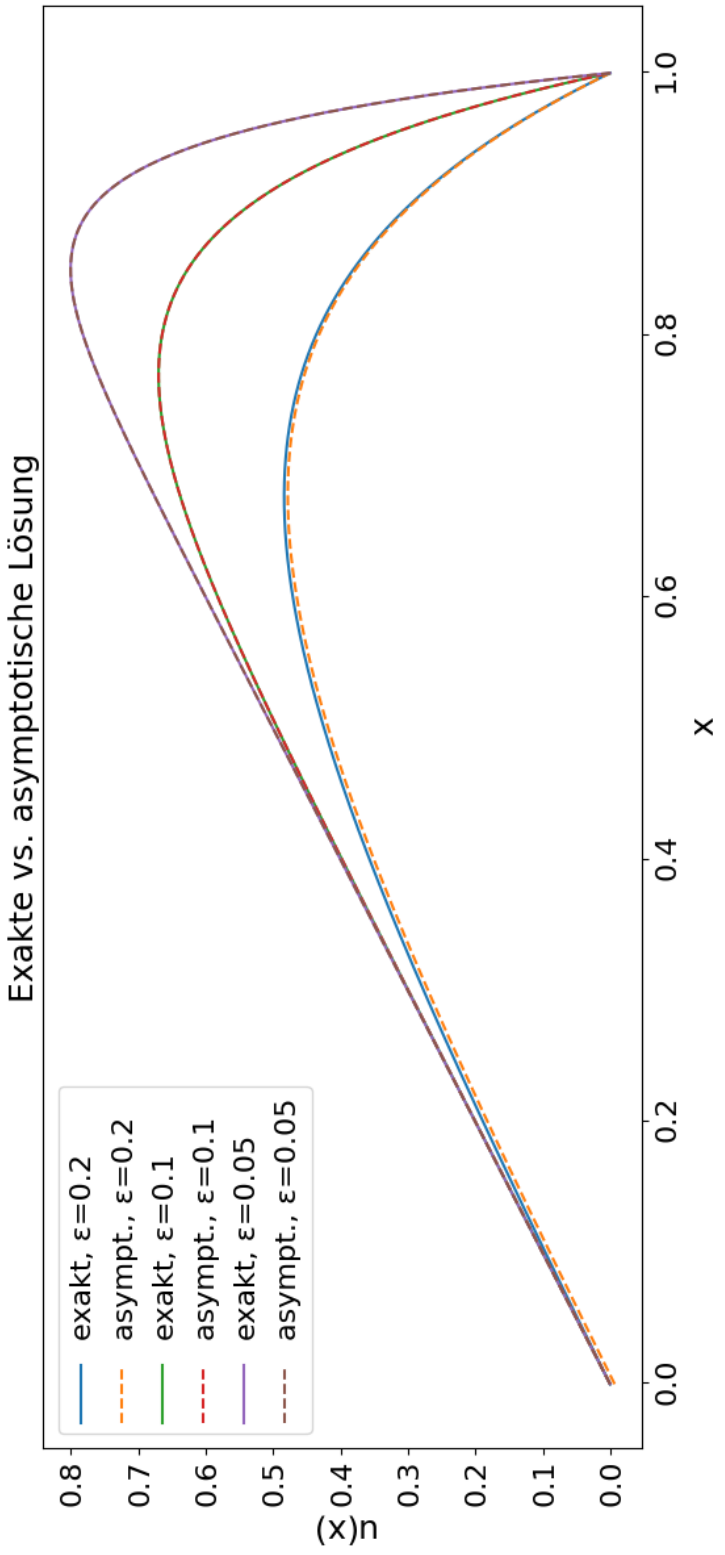

Abbildung 2.1: Exakte und asymptotische Lösung für verschiedene Werte von ε.

Wir betrachten zusätzlich den Fehler

$$|u_{\text{exakt}}(x) - u_{\text{as}}(x)|$$

auf logarithmischer Skala.

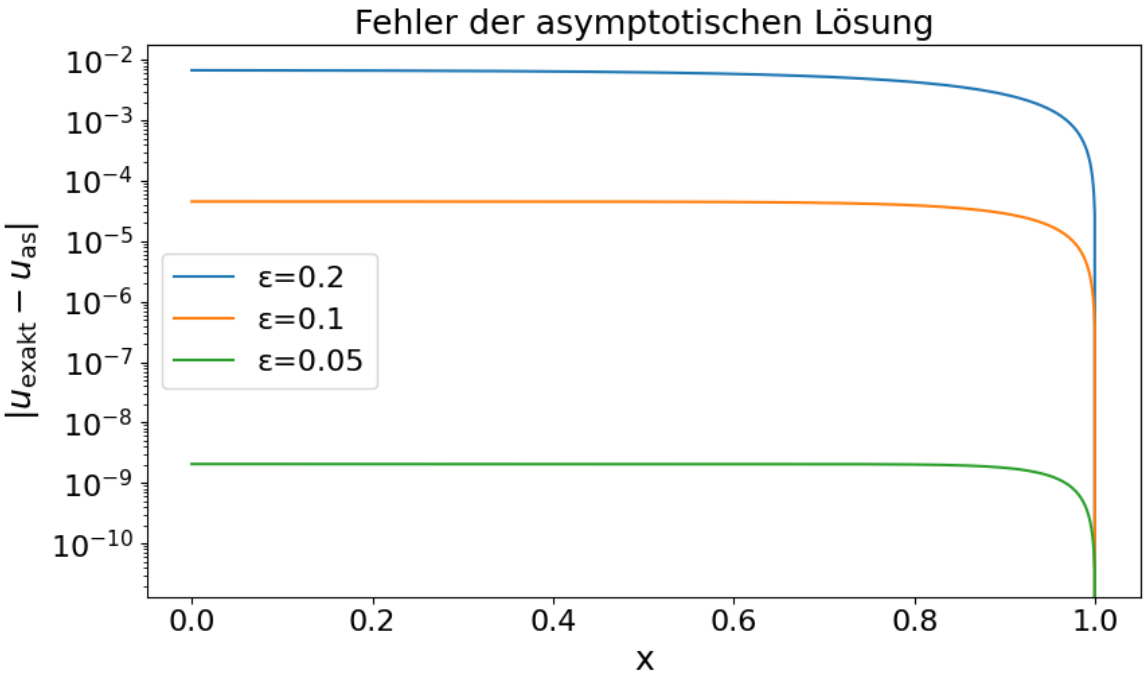

Abbildung 2.2: Absoluter Fehler der asymptotischen Lösung auf logarithmischer Skala.

Kapitel 3

Finite-Differenzen-Methoden

Die zentrale Idee der Finite-Differenzen-Methoden lautet wie folgt: Wir wollen das Gebiet, auf dem die Differentialgleichung definiert ist, diskretisieren, d. h. das Gebiet in kleine Abschnitte, in diesem Falle Gitterzellen, aufteilen. Auf jedem Gitterpunkt werden die Ableitungen durch Näherungen ersetzt, unter Nutzung der Werte der benachbarten Gitterpunkte. Man erhält ein Gleichungssystem für die Werte in den Gitterpunkten. Dieses wird anschließend gelöst. Damit erhält man eine Näherung für die Lösung des Zwei-Punkt-Randwertproblems.

3.1 Bestimmung der Näherungen

3.1.1 Numerische Differenzierung

Definition 3.1.1 (Ableitung von u) [6, p. 2]
Die Ableitung von u ist definiert als

$$u'(x) = \lim_{h \to 0} \frac{u(x+h) - u(x)}{h}. \tag{3.1}$$

Definition 3.1.2 (Differenzenoperatoren) [6, p. 2]
Bei ausreichend kleinem h kann die rechte Seite ohne den Grenzwert als Näherung für $u'(x)$ genutzt werden. Häufig genutzte Näherungen, sogenannte Differenzenoperatoren, lauten:

$$\begin{aligned}
\textbf{Vorwärtsdifferenz:} \quad & \partial^+ u(x) := \frac{u(x+h) - u(x)}{h}, \\
\textbf{Rückwärtsdifferenz:} \quad & \partial^- u(x) := \frac{u(x) - u(x-h)}{h}, \\
\textbf{Zentrale Differenz:} \quad & \partial^\circ u(x) := \frac{u(x+h) - u(x-h)}{2h} = \frac{1}{2}\left(\partial^+ u(x) + \partial^- u(x)\right).
\end{aligned}$$

Eine häufig verwendete Näherung der **zweiten Ableitung** ist:

$$\partial^+ \partial^- u(x) := \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}.$$

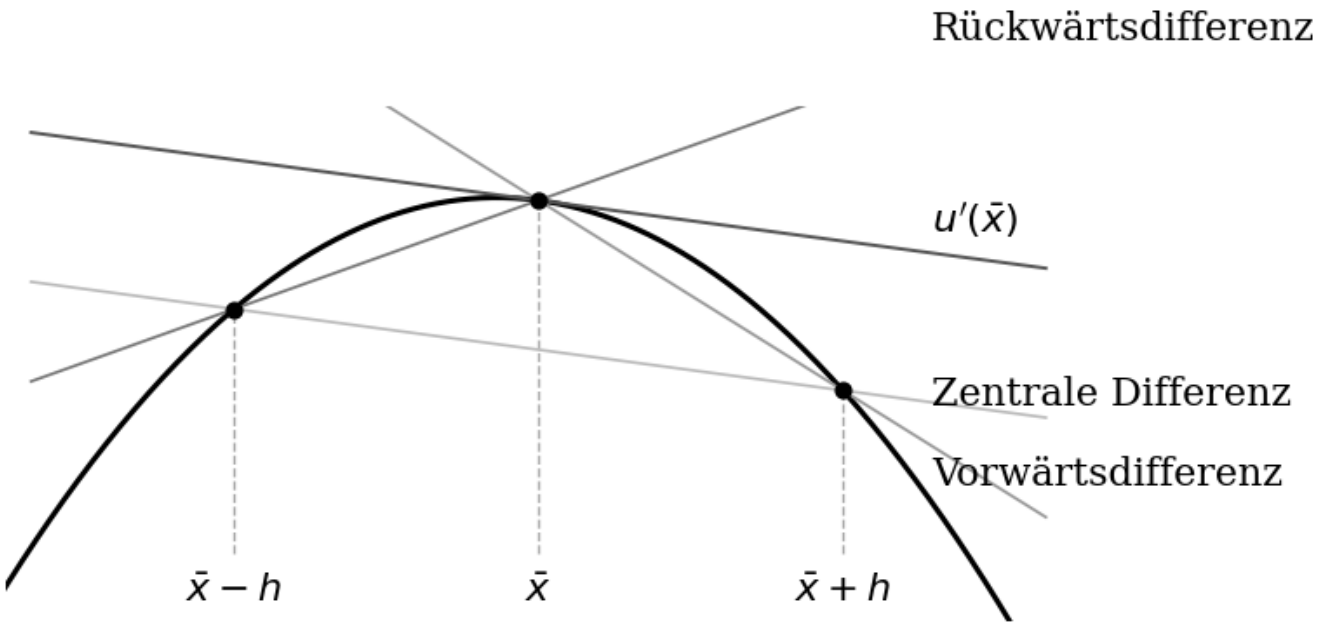

Abbildung 3.1: Verschiedene Näherungen an die Ableitung $u'(\bar{x})$, dargestellt über die Steigungen zugehöriger Sekanten, in Anlehnung an [7, p. 4].

3.1.2 Fehleranalyse

Definition 3.1.3 (Gitterfunktion) [1, p. 500]
Ein Vektor $\vec{v}_h = (v_0, \ldots, v_N)^T \in \mathbb{R}^{N+1}$, der jedem Gitterpunkt einen Wert zuordnet, heißt **Gitterfunktion**.
Die Restriktion einer Funktion $v \in C([0,1])$ auf eine Gitterfunktion wird definiert als:

$$R_h v := \big(v(x_0), v(x_1), \ldots, v(x_N)\big).$$

Definition 3.1.4 (Diskrete Maximumsnorm im Raum der Gitterfunktionen) [7, p. 16]
Sei $w_h = \{x_i \mid i = 0, \ldots, N\}$ ein Gitter, und sei $u : w_h \to \mathbb{R}$ eine Gitterfunktion.
Dann ist die **diskrete Maximumsnorm** von u definiert als:

$$\|u\|_{\infty,h} = \max_{x \in w_h} |u(x)|.$$

Definition 3.1.5 (Diskrete L^2-Norm)
Sei $u : w_h \to \mathbb{R}$ eine Gitterfunktion und $w_h = \{x_i \mid i = 0, \ldots, N\}$ ein Gitter mit Schrittweite $h = \frac{1}{N}$.
Dann ist die **diskrete L^2-Norm** von u definiert als:

$$\|u\|_{2,h} = \left(h \sum_{i=0}^{N} |u(x_i)|^2 \right)^{1/2}.$$

Definition 3.1.6 (Konsistenz von Differenzenoperatoren) [1, p. 501]
Sei L ein Differentialoperator. Der Finite-Differenzenoperator $L_h : \mathbb{R}^n \to \mathbb{R}^n$ heißt **zu L konsistent mit Ordnung** k, wenn für alle ausreichend glatte Funktionen u gilt:

$$\max_{1 \le i \le n} |(Lu)(x_i) - (L_h R_h u)_i| =: \|Lu - L_h R_h u\|_{\infty,h} = \mathcal{O}(h^k).$$

Die Konsistenzordnung eines Finite-Differenzenoperators wird in der Regel mit der Taylor-Entwicklung bestimmt.

Bemerkung 3.1.7 (Fehler der Standard-Differenzenoperatoren) [6, p. 4-5]
Zur Fehleranalyse der Näherungen nutzt man eine Taylor-Entwicklung des Fehlers um x. Wir nehmen an, dass die Funktionen ausreichend glatt sind.

Vorwärtsdifferenz:

$$e(x,h) = u'(x) - \frac{u(x+h)-u(x)}{h} = u'(x) - \frac{u(x)+u'(x)h+\frac{1}{2}u''(\zeta)h^2 - u(x)}{h}$$

$$= -\frac{h}{2}u''(\zeta), \quad \zeta \in (x, x+h).$$

Rückwärtsdifferenz:

$$e(x,h) = u'(x) - \frac{u(x)-u(x-h)}{h} = \frac{h}{2}u''(\zeta), \quad \zeta \in (x-h, x).$$

Zentrale Differenz:

$$e(x,h) = u'(x) - \frac{u(x+h)-u(x-h)}{2h} = -\frac{h^2}{6}u'''(\zeta), \quad \zeta \in (x-h, x+h).$$

Näherung der zweiten Ableitung:

$$e(x,h) = \frac{u(x+h)-2u(x)+u(x-h)}{h^2} - u''(x) = \frac{h^2}{12}u^{(4)}(\zeta), \quad \zeta \in (x-h, x+h).$$

Bemerkung 3.1.8 (Konsistenz der Standard-Differenzenoperatoren) [4, p. 21]
Wir können für die Näherungen erkennen:

$$\begin{aligned}
\partial^+ u(x) &= u'(x) + \mathcal{O}(h), \\
\partial^- u(x) &= u'(x) + \mathcal{O}(h), \\
\partial^\circ u(x) &= u'(x) + \mathcal{O}(h^2), \\
\partial^+\partial^- u(x) &= u''(x) + \mathcal{O}(h^2).
\end{aligned}$$

Die Differenzenoperatoren $\partial^+ u(x)$ und $\partial^- u(x)$ sind damit konsistent zu $L = \frac{d}{dx}$ mit Ordnung 1, $\partial^\circ u(x)$ mit Ordnung 2. Der Operator $\partial^+\partial^- u(x)$ ist von zweiter Ordnung konsistent zu $L = \frac{d^2}{dx^2}$.

3.2 Zentrales Differenzenverfahren

Dieser Abschnitt folgt [6, p. 10-11].
Wir betrachten erneut das in Definition 2.1.4 eingeführte Standardproblem

$$Lu := -u'' + bu' + \sigma u = f \quad \text{in} \quad \Omega = (0,1), \tag{3.2}$$

mit Randbedingungen

$$u(0) = u(1) = 0. \tag{3.3}$$

1. Diskretisierung des Intervalls
Auf dem Intervall $[c, d] = [0, 1]$ wählen wir ein $N \in \mathbb{N}$ und definieren die Schrittweite

$$\frac{d-c}{N} = \frac{1}{N} =: h.$$

Die Stützstellen lauten

$$x_i = ih, \quad i \in \{0, \ldots, N\},$$

wir bezeichnen das Gitter mit

$$w_h = \{\, x_i \mid i = 0, \ldots, N \,\}.$$

2. Ersetzen der Ableitungen
Für jede innere Stützstelle x_i, $i = 1, ..., N-1$, ersetzen wir die Ableitungen durch Näherungen aus Definition 3.1.2. Wir setzen

$$-\varepsilon\,(\partial^+\partial^- u)(x_i) + b(x_i)\,(\partial^\circ u)(x_i) + \sigma(x_i)\,u(x_i) + O(h^2) = f(x_i).$$

Dadurch erhalten wir

$$\begin{aligned} &-\varepsilon\,\frac{u(x_i+h) - 2u(x_i) + u(x_i-h)}{h^2} \;+\; b(x_i)\,\frac{u(x_i+h) - u(x_i-h)}{2h} \\ &+\; \sigma(x_i)\,u(x_i) \;+\; \mathcal{O}(h^2) \;=\; f(x_i). \end{aligned} \tag{3.4}$$

3. Numerische Approximation
Wir definieren

$$u_i \approx u(x_i), \qquad i = 0, \ldots, N,$$

als numerische Approximation der exakten Lösung an den Gitterpunkten. Durch Vernachlässigung des Fehlerterms $\mathcal{O}(h^2)$ erhalten wir das zentrale Differenzensystem

$$-\varepsilon\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + b(x_i)\frac{u_{i+1} - u_{i-1}}{2h} + \sigma(x_i)u_i = f(x_i). \tag{3.5}$$

4. Randbedingungen
Die Randwerte werden durch

$$u_0 = u(0) = 0, \qquad u_N = u(1) = 0$$

direkt in das Gleichungssystem übertragen.

5. Lineares Gleichungssystem
Nach Multiplikation von (3.5) mit h^2 erhält man

$$(-u_{i+1} + 2u_i - u_{i-1})\varepsilon \;+\; \frac{h\,b(x_i)}{2}\,(u_{i+1} - u_{i-1}) \;+\; h^2\sigma(x_i)u_i = h^2 f(x_i). \tag{3.6}$$

Wir definieren die Koeffizienten

$$g_i = -\varepsilon - \frac{h}{2}\,b(x_i), \qquad h_i = 2\varepsilon + h^2\sigma(x_i), \qquad j_i = -\varepsilon + \frac{h}{2}\,b(x_i).$$

Damit ergibt sich das lineare System $Au = F$ der Größe $(N+1) \times (N+1)$:

$$A = \begin{pmatrix} 1 & 0 & \cdots & & & & \\ g_1 & h_1 & j_1 & \cdots & & & \\ 0 & g_2 & h_2 & j_2 & \cdots & & \\ 0 & 0 & g_3 & \ddots & \ddots & & \\ \vdots & \vdots & 0 & \ddots & & & \\ & & & & g_{N-1} & h_{N-1} & j_{N-1} \\ & & & & & 0 & 1 \end{pmatrix}, \quad u = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}, \quad F = \begin{pmatrix} 0 \\ h^2 f(x_1) \\ \vdots \\ h^2 f(x_{N-1}) \\ 0 \end{pmatrix}. \tag{3.7}$$

Beispiel 3.2.1 (Finite-Differenzen-Methode) In diesem Beispiel wird die Finite-Differenzen-Methode zur Lösung der Differentialgleichung

$$-u''(x) + b(x)u'(x) + \sigma(x)u(x) = f(x), \quad x \in (0,1),$$

mit Randbedingungen $u(0) = 0$ und $u(1) = 0$ genutzt.
Wir wählen die Funktionen $b(x) = 0$, $\sigma(x) = 0$ und $f(x) = \pi^2 \sin(\pi x)$ mit bekannter exakter Lösung

$$u(x) = \sin(\pi x).$$

Das Intervall $[0,1]$ wird in N gleich große Teilintervalle unterteilt, und für $N = 4, 8, 16$ wird das resultierende lineare Gleichungssystem numerisch gelöst.
Wir vergleichen die numerischen Lösungen mit den exakten und berechnen die Fehler in der L^2-Norm und in der Maximumsnorm.

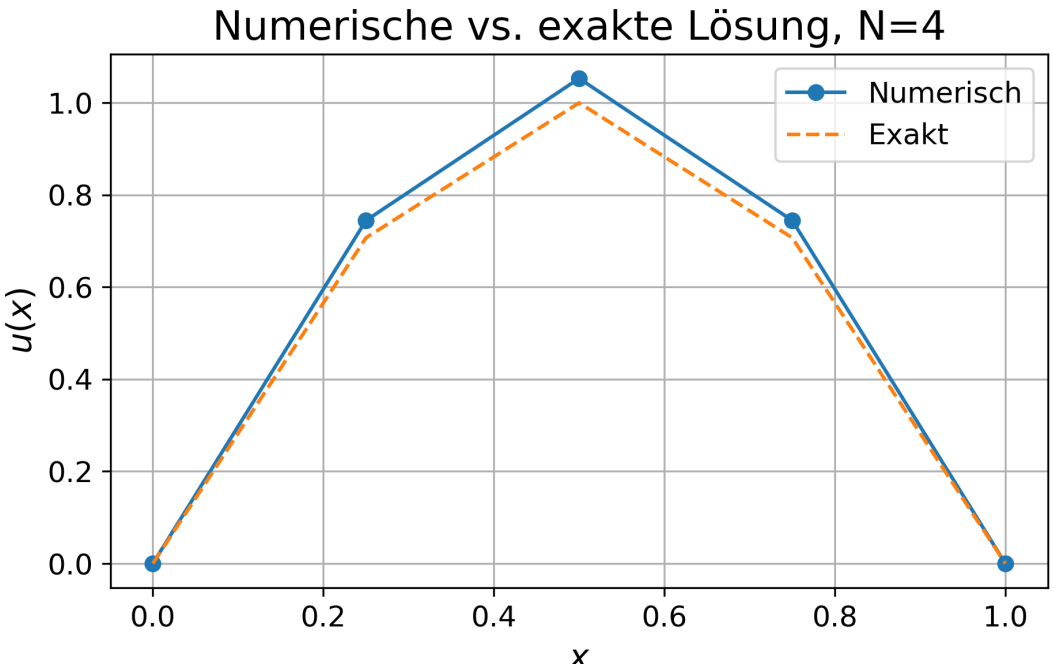

Abbildung 3.2: Numerische Lösung für $N = 4$

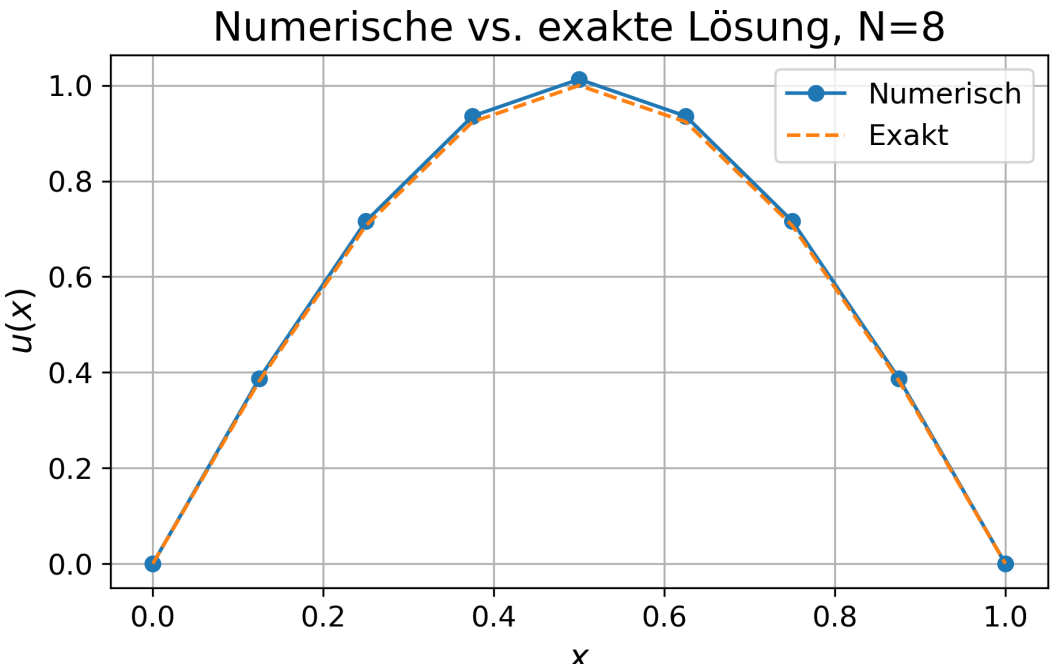

Abbildung 3.3: Numerische Lösung für $N = 8$

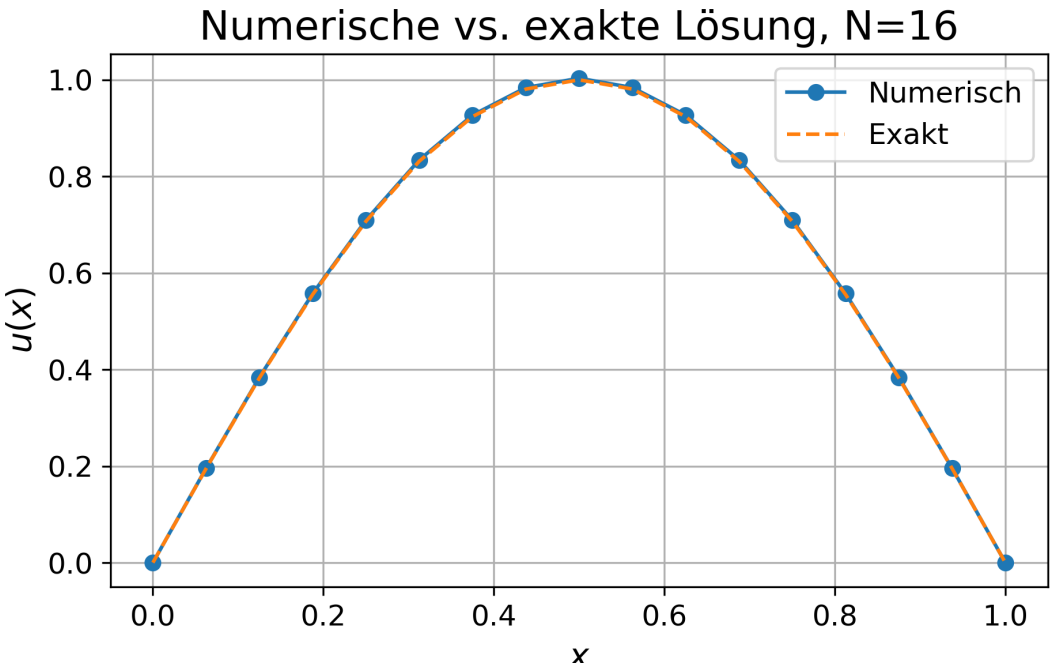

Abbildung 3.4: Numerische Lösung für $N = 16$

Mit zunehmender Gitterfeinheit nähert sich die numerische Lösung immer mehr der exakten Lösung an. In den berechneten Fehlernormen kann man das ebenfalls erkennen:

N	$\|e\|_{2,h}$	$\|e\|_{\infty,h}$
4	$3.75 \cdot 10^{-2}$	$5.30 \cdot 10^{-2}$
8	$9.16 \cdot 10^{-3}$	$1.30 \cdot 10^{-2}$
16	$2.28 \cdot 10^{-3}$	$3.22 \cdot 10^{-3}$

Tabelle 3.1: Diskrete Fehlernormen für das zentrale Differenzenverfahren.

3.3 Fehleranalyse

3.3.1 Globaler Fehler

Definition 3.3.1 (Globaler Fehler) [8, p. 22]
Seien $u = (u_0, \ldots, u_N)^\top$ die Näherungslösung aus Kapitel 3.2 und $u(x_i)$ die exakte Lösung ausgewertet an den Gitterpunkten x_i. Dann ist der globale Fehlervektor definiert als

$$E = (u_0 - u(x_0),\ \ldots,\ u_N - u(x_N))^\top.$$

Wir suchen eine obere Schranke für den Fehler, häufig gemessen in

- der 1 - Norm $\|E\|_1 = \sum_i h_i |e_i|$, wobei $h_i = x_{i+1} - x_i$,
- der 2 - Norm $\|E\|_2 = \left(\sum_{i=0}^{N} h_i |e_i|^2 \right)^{1/2}$,
- der Maximumsnorm.

Definition 3.3.2 (Genauigkeit von Ordnung p) [8, p. 22]
Wir sagen, dass eine Finite-Differenzen-Methode **Genauigkeit von Ordnung** p hat, wenn

$$\|E\| \leq Ch^p, \quad p > 0.$$

Definition 3.3.3 (Konvergenz) [8, p. 22]
Wir betrachten die Normen aus Definition 3.3.1. Ein Finite-Differenzen-Verfahren heißt konvergent in einer dieser Normen, wenn

$$\lim_{h \to 0} \|E\| = 0.$$

3.3.2 Lokaler Fehler

Der lokale Diskretisierungsfehler bezeichnet die Differenz zwischen der ursprünglichen Differentialgleichung und der Finite-Differenzen-Diskretisierung an den Gitterpunkten. Er misst, wie gut die Finite-Differenzen-Diskretisierung die Differentialgleichung approximiert (vgl. [8, p. 22]).

Definition 3.3.4 (Konsistenz eines Differenzenschemas) [1, p. 501]
Wir betrachten nun die lineare partielle Differentialgleichung $Lu = f$ und eine lineare Finite-Differenzen-Näherung der Form

$$L_h u_h = R_h(Lu) = R_h f = f_h$$

auf einem äquidistanten Gitter. Dieses Schema heißt konsistent von Ordnung k in der diskreten Maximumsnorm, wenn für alle ausreichend glatte v gilt:

$$\|L_h R_h v - R_h(Lv)\|_{\infty,h} = \mathcal{O}(h^k),$$

wobei $k > 0$ unabhängig von h ist.

Definition 3.3.5 (Stabilität) [1, p. 502]
Ein Finite-Differenzen-Schema (bzw. ein Finite-Differenzenoperator) heißt stabil in der diskreten Maximumsnorm, wenn es eine von h unabhängige Konstante C_S gibt, sodass

$$\|v_h\|_{\infty,h} \leq C_S \|L_h v_h\|_{\infty,h}$$

für alle Gitterfunktionen v_h gilt.

Definition 3.3.6 (Konvergenz eines Differenzenschemas) [1, p. 502]
Das Differenzenverfahren

$$L_h u_h = R_h(Lu) = R_h f = f_h$$

heißt konvergent von Ordnung k in der diskreten Maximumsnorm, wenn es eine von h unabhängige positive Konstante k gibt, so dass

$$\|R_h u - u_h\|_{\infty,h} = \mathcal{O}(h^k).$$

Theorem 3.3.7 (Konsistenz + Stabilität $\Rightarrow$ Konvergenz) [1, p. 502]
Ein konsistentes und stabiles Finite-Differenzen-Schema ist konvergent. Die Ordnungen der Konvergenz und Konsistenz sind identisch.

Kapitel 4

Konvektions-dominanter Fall

Im Folgenden wird untersucht, wie sich die Lösung für kleine Werte von ε verhält. Wird ε sehr klein und ist $b(x)$ vergleichsweise groß, so dominiert der Term $b(x)u'(x)$ gegenüber dem Term $-\varepsilon u''(x)$. In diesem Fall treten sogenannte *Grenzschichten* auf, also Bereiche nahe den Randpunkten, in denen die Lösung auf sehr kleinen Intervallen starke Änderungen zeigt.

4.1 Grenzschicht

Definition 4.1.1 (Grenzschicht) [10, p. 4]
Sei $v = v(x, \varepsilon)$ für $(x, \varepsilon) \in [0, 1] \times (0, 1]$. Es wird angenommen, dass

$$v'(x, \varepsilon) := \frac{\partial v(x, \varepsilon)}{\partial x}$$

für alle $(x, \varepsilon) \in [0, 1] \times (0, 1]$ existiert. Wir sagen, dass v für $\varepsilon \to 0^+$ eine Grenzschicht an einem Punkt $z \in [0, 1]$ besitzt, wenn die folgenden Bedingungen erfüllt sind:

1. $\lim_{\varepsilon \to 0^+} v'(z, \varepsilon) = \pm\infty$,
2. $\lim_{\varepsilon \to 0^+} v'(x, \varepsilon)$ existiert und ist endlich für alle Punkte$x \in [0, 1]$, die

   $$0 < |x - z| < k$$

   für eine positive Konstante k erfüllen, wobei k von z, aber nicht von ε abhängen darf.

Beispiel 4.1.2 (Beispiel mit konstanten Koeffizienten b und σ)
Wir betrachten die Differentialgleichung

$$-\varepsilon u''(x) + u'(x) = 0, \qquad x \in (0, 1), \tag{4.1}$$

mit Randbedingungen

$$u(0) = 0, \qquad u(1) = 1.$$

Dies entspricht dem Fall der Modellgleichung mit $b(x) = 1$, $\sigma = 0$. Da $\varepsilon > 0$, erhalten wir

$$u''(x) - \frac{1}{\varepsilon} u'(x) = 0. \tag{4.2}$$

Das ist eine homogene lineare Differentialgleichung 2. Ordnung. Durch Substitution $v(x) = u'(x)$ und damit $v'(x) = u''(x)$ erhalten wir eine Differentialgleichung 1. Ordnung:

$$v'(x) - \frac{1}{\varepsilon} v(x) = 0 \;\Leftrightarrow\; v'(x) = \frac{1}{\varepsilon} v(x). \tag{4.3}$$

Diese Gleichung besitzt eine Lösung der Form

$$v(x) = \alpha e^{\frac{1}{\varepsilon}x} \tag{4.4}$$

für ein $\alpha \in \mathbb{R}$.
Es wird eine Rücksubstitution durchgeführt. Sei $C \in \mathbb{R}$. Dann gilt

$$\begin{aligned} u(x) &= \int v(x)\,dx = \int \alpha e^{\frac{1}{\varepsilon}x}\,dx = \alpha \int e^{\frac{1}{\varepsilon}x}\,dx \\ &= \alpha \left(\varepsilon e^{\frac{1}{\varepsilon}x} + C \right) = \alpha' e^{\frac{1}{\varepsilon}x} + C', \end{aligned} \tag{4.5}$$

wobei $\alpha' := \alpha\varepsilon$ und $C' := \alpha C$ erneut Konstanten sind. Jetzt können wir unsere Randbedingungen nutzen, um C' und α' zu bestimmen.

1. Aus der ersten Randbedingung folgt

$$u(0) = 0 \Rightarrow \alpha' e^0 + C' = 0 \Rightarrow \alpha' = -C'. \tag{4.6}$$

2. Mit (4.6) und der zweiten Randbedingung erhalten wir

$$\begin{aligned} u(1) = 1 &\Rightarrow \alpha' e^{\frac{1}{\varepsilon}} + C' = 1 \\ &\Rightarrow \alpha' e^{\frac{1}{\varepsilon}} - \alpha' = 1 \\ &\overset{\alpha' \neq 0}{\Rightarrow} e^{\frac{1}{\varepsilon}} - 1 = \frac{1}{\alpha'} \\ &\Rightarrow \alpha' = \frac{1}{e^{\frac{1}{\varepsilon}} - 1} \Rightarrow C' = -\frac{1}{e^{\frac{1}{\varepsilon}} - 1}. \end{aligned}$$

Damit folgt insgesamt:

$$u(x) = \frac{e^{\frac{1}{\varepsilon}x} - 1}{e^{\frac{1}{\varepsilon}} - 1}. \tag{4.7}$$

Grafik

Die folgenden Abbildungen illustrieren das Verhalten der Lösung für kleine Werte von ε.

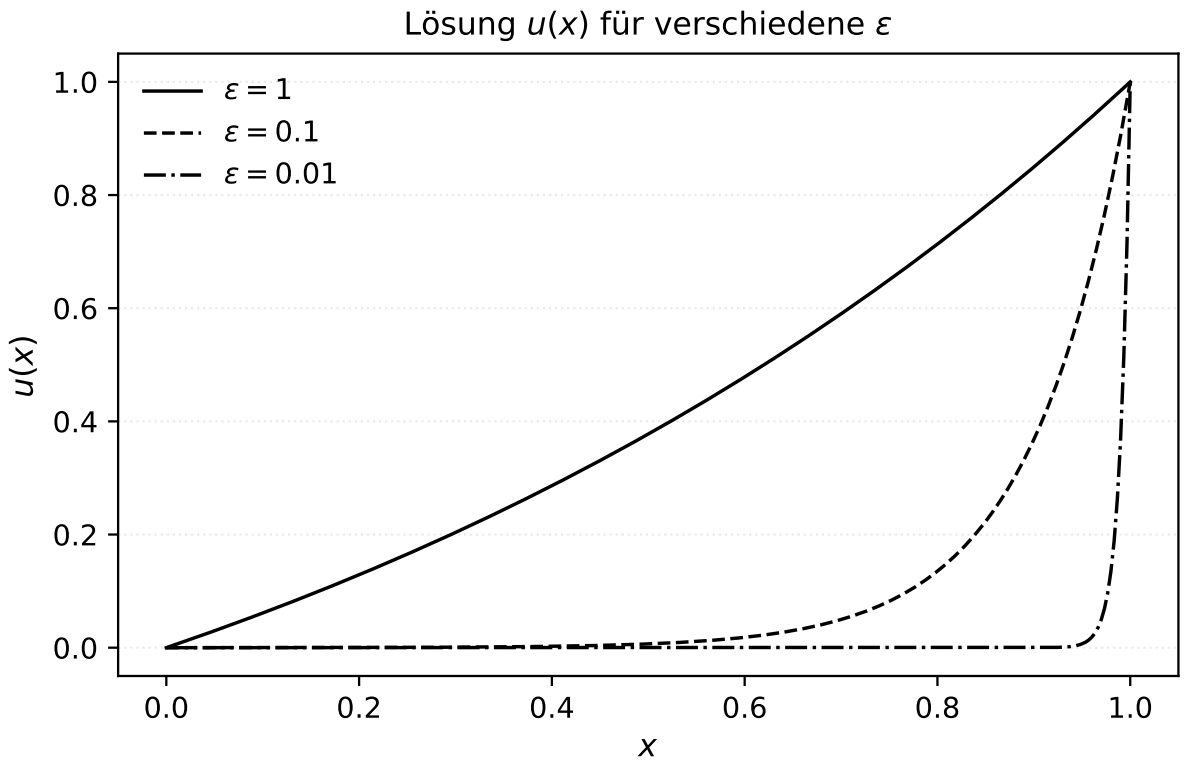

Abbildung 4.1: Lösungsverlauf $u(x)$ für verschiedene Werte von ε. Für kleine ε erkennt man eine deutliche Grenzschicht nahe $x = 1$.

Wir betrachten noch einmal explizit die Grenzschicht:

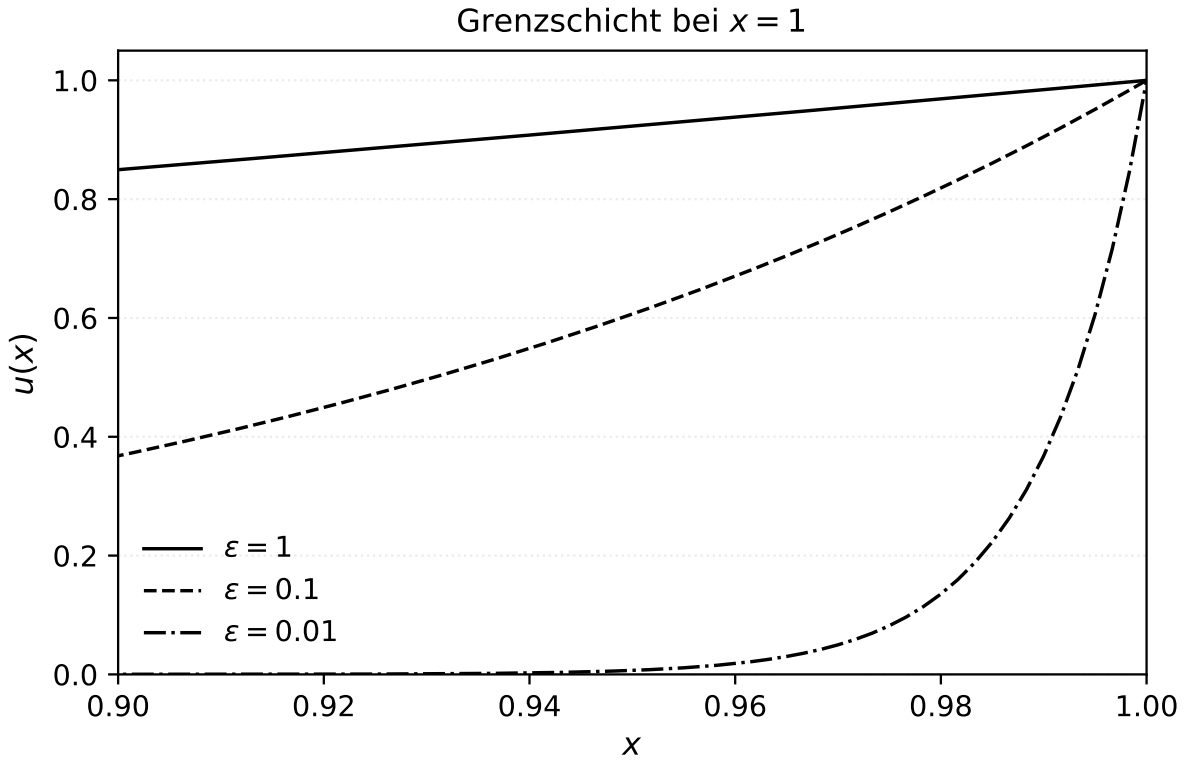

Abbildung 4.2: Vergrößerung der Grenzschichtregion bei $x = 1$ für kleine Werte von ε.

Beispiel 4.1.3 (Beispiel mit nicht-konstanten Koeffizienten b und σ)

Betrachtet wird das Randwertproblem

$$-\varepsilon u''(x) + b(x)\,u'(x) + \sigma(x)\,u(x) = f(x), \qquad x \in (0,1),$$

mit Randbedingungen

$$u(0) = 0, \qquad u(1) = 1.$$

1. Als exakte Lösung wählen wir $u(x) = x$. Zudem setzen wir

$$b(x) = 1 + x, \qquad \sigma(x) = 1, \qquad \varepsilon = 0{,}1.$$

2. Aus $u'(x) = 1$ und $u''(x) = 0$ folgt für die rechte Seite

$$\begin{aligned} f(x) &= -\varepsilon u''(x) + b(x)u'(x) + \sigma(x)u(x) \\ &= -0{,}1 \cdot 0 + (1+x) \cdot 1 + x \\ &= 1 + 2x. \end{aligned} \tag{4.8}$$

3. Damit ergibt sich die vollständige Aufgabe

$$-0{,}1\,u''(x) + (1+x)\,u'(x) + u(x) = 1 + 2x, \qquad u(0) = 0,\ u(1) = 1,$$

   für welche $u(x) = x$ eine Lösung ist.

4. **Eindeutigkeit der Lösung**
   Seien u und v zwei beliebige Lösungen des Randwertproblems, und es sei

$$w(x) := u(x) - v(x).$$

   Dann erfüllt w die Gleichung

$$-\varepsilon w''(x) + b(x)\,w'(x) + \sigma(x)\,w(x) = 0, \qquad w(0) = w(1) = 0.$$

   Multiplikation mit $w(x)$ und Integration über $(0,1)$ liefert

$$\int_0^1 (-\varepsilon w'' + bw' + \sigma w)\,w\,dx = 0.$$

   Erster Term
   Partielle Integration ergibt

$$\int_0^1 -\varepsilon w''(x) w(x)\,dx = \varepsilon \int_0^1 (w'(x))^2\,dx.$$

   Zweiter Term
   Aus der Produktregel folgt

$$w'(x)w(x) = \tfrac{1}{2}(w^2(x))',$$

sodass

$$\int_0^1 b(x)w'(x)w(x)\,dx = -\frac{1}{2}\int_0^1 b'(x)w^2(x)\,dx.$$

Zusammenfassung
Daraus folgt

$$\varepsilon\int_0^1 (w')^2\,dx - \frac{1}{2}\int_0^1 b'(x)w^2\,dx + \int_0^1 \sigma(x)w^2\,dx = 0.$$

Die beiden letzten Integrale lassen sich zusammenfassen. Wir erhalten

$$\varepsilon\int_0^1 (w')^2\,dx + \int_0^1 \left(\sigma(x) - \tfrac{1}{2}b'(x)\right)w^2(x)\,dx = 0.$$

Für die hier verwendeten Koeffizienten gilt

$$b'(x) = 1, \qquad \sigma(x) = 1,$$

sodass

$$\sigma - \tfrac{1}{2}b' = \tfrac{1}{2} > 0.$$

Damit erhält man

$$\varepsilon\int_0^1 (w')^2\,dx + \frac{1}{2}\int_0^1 w^2(x)\,dx = 0.$$

Beide Integrale sind nichtnegativ, sodass sie also beide verschwinden müssen. Daraus folgt zunächst $w'(x) \equiv 0$, also ist w konstant; aufgrund der Randbedingungen $w(0) = w(1) = 0$ ergibt sich schließlich $w \equiv 0$.

Somit ist die Lösung des Randwertproblems eindeutig.

4.2 Supremumsnorm

Die Supremumsnorm einer Funktion $f\colon D \to \mathbb{R}$ ist definiert als

$$\|f\|_\infty := \sup_{x\in D} |f(x)|,$$

vgl. [10, p. 7].

4.3 Ausblick: Konvektions-dominanter Fall

Wir betrachten in Zukunft den Fall $\|b\|_{L^\infty(\Omega)} \gg \varepsilon$, d. h. Konvektion dominiert gegenüber Diffusion.

Kapitel 5

Stabile und gleichmäßig konvergente Verfahren

5.1 M-Matrizen

Definition 5.1.1 (Natürliche Ordnung von Vektoren und Matrizen) [4, p. 24]
Seien $x, y \in \mathbb{R}^n$. Es gilt

$$x \leq y \quad :\Leftrightarrow \quad x_i \leq y_i \text{ für alle } i = 1, \ldots, n.$$

Analog gilt $x \geq \mathbf{1}$, falls $x_i \geq 1$ für alle $i = 1, \ldots, n$.
Für eine Matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ gilt

$$A \geq 0 \quad :\Leftrightarrow \quad a_{ij} \geq 0 \text{ für alle } i, j = 1, \ldots, n.$$

Definition 5.1.2 (Invers-monotone Matrix) [4, p. 24]
Eine Matrix A heißt invers–monoton, falls A^{-1} existiert und

$$A^{-1} \geq 0.$$

Lemma 5.1.3 (Diskretes Vergleichsprinzip) [4, p. 24]
Sei $A \in \mathbb{R}^{n \times n}$ invers–monoton. Dann gilt für beliebige Vektoren $v, w \in \mathbb{R}^n$:

$$Av \leq Aw \quad \Rightarrow \quad v \leq w.$$

Beweis.

$$Av \leq Aw \Rightarrow A(v - w) \leq 0$$

$$\overset{\cdot A^{-1}}{\Rightarrow} v - w = A^{-1}(A(v - w)) \leq 0.$$

A ist invers-monoton. Daraus folgt

$$v \leq w.$$

□

Definition 5.1.4 (M-Matrix) [10, p. 45]
Eine quadratische Matrix $A = (A_{ij})$ wird M-Matrix genannt, falls

1. $A_{ij} \leq 0$ für alle $i \neq j$ und
2. A^{-1} existiert mit $A^{-1} \geq 0$.

Die zweite Bedingung für eine M-Matrix - dass A^{-1} existiert mit $(A^{-1})_{ij} \geq 0$ für alle i und j - ist in der Praxis nicht leicht zu überprüfen. Man nutzt daher alternative Kriterien, die leichter verifizierbar sind.

Definition 5.1.5 (Strikte Diagonaldominanz) [10, p. 45]
Eine quadratische Matrix $A = (A_{ij})$ heißt strikt diagonaldominant, wenn für alle i gilt:

$$A_{ii} > \sum_{j \neq i} |A_{ij}|.$$

Lemma 5.1.6 [10, p. 45]
Sei $A = (A_{ij})$ eine quadratische Matrix, für die gilt:

$$A_{ij} \leq 0 \quad \text{für alle } i \neq j.$$

Sei A zudem strikt diagonaldominant. Damit erfüllen alle Diagonaleinträge

$$A_{ii} > 0.$$

Dann existiert A^{-1} und es gilt

$$(A^{-1})_{ij} \geq 0 \quad \text{für alle } i, j.$$

Satz 5.1.7 (M-Matrix-Kriterium) [4, p. 24]
Sei $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ mit

$$a_{ij} \leq 0 \quad \text{für } i \neq j.$$

Dann ist A genau dann eine M-Matrix, wenn ein Vektor

$$w \in \mathbb{R}^n, \quad w > 0,$$

existiert, so dass

$$Aw > 0.$$

In diesem Fall gilt für die Zeilensummennorm

$$\|A^{-1}\|_\infty \leq \frac{\|w\|_{\infty,d}}{\min\limits_k (Aw)_k}. \tag{5.1}$$

Der Vektor w wird majorisierendes Element genannt.

Bemerkung 5.1.8 (Zum M-Matrix-Kriterium) [1, p. 502-503]
Es sei eine Familie von Finite-Differenzen-Gittern über einem Gebiet Ω mit Gitterparametern $h > 0$ gegeben, wobei $h \to 0$ gelte. Angenommen, es existiere ein $H > 0$, so dass das zugehörige Finite-Differenzen-Verfahren für alle $h \leq H$ zu einer M-Matrix A führt. Für jedes $h \leq H$ sei $w_h \in \mathbb{R}^n$ ein majorisierendes Element, d. h.

$$w_h > 0 \quad \text{und} \quad Aw_h > 0.$$

Dann ist das Finite-Differenzen-Verfahren für alle $h \leq H$ stabil und es gilt

$$C_S \leq \sup_{h \in (0,H]} \frac{\|w_h\|_{\infty,h}}{\min\limits_{j=1,\ldots,n} (Aw_h)_j}, \tag{5.2}$$

sofern der Ausdruck auf der rechten Seite endlich ist.

Beweis. Der Beweis folgt Barrenechea, John und Knobloch [1]. □

Lemma 5.1.9 (Stabilität des zentralen Differenzenschemas für hinreichend feine Gitter) [4, p. 25]
Für $\varepsilon = 1$ und hinreichend kleine Gitterweiten h ist das zentrale Differenzenschema 3.5 für das Randwertproblem (3.2), (3.3) in der diskreten Maximumsnorm stabil. Die zugehörige Koeffizientenmatrix ist eine M–Matrix.

Lemma 5.1.10 (Diskretes Maximumprinzip) [10, p. 46]
Sei A eine M-Matrix. Gilt für einen Vektor $w \in \mathbb{R}^n$

$$Aw \geq 0,$$

so folgt daraus

$$w \geq 0.$$

Beweis. Da A eine M-Matrix ist, existiert A^{-1} und es gilt $A^{-1} \geq 0$. Damit folgt

$$w = A^{-1}(Aw) \geq 0,$$

da sowohl $A^{-1} \geq 0$ als auch $Aw \geq 0$. □

Lemma 5.1.11 (Diskrete Barrierefunktion) [10, p. 46]
Sei A eine M-Matrix. Wenn w, z Vektoren sind mit

$$|Aw| \leq Az,$$

dann gilt

$$|w| \leq z.$$

Beweis. Sei A eine M-Matrix und $|Aw| \leq Az$,

$$\Rightarrow -Az \leq Aw \leq Az.$$

Daraus ergeben sich folgende Gleichungen:

$$Az - Aw \geq 0 \quad \text{und} \quad Az + Aw \geq 0.$$

Wir nutzen Lemma 5.1.10:

- $A(z - w) = Az - Aw \geq 0 \overset{5.1.10}{\Longrightarrow} z - w \geq 0 \Rightarrow w \leq z.$
- $A(z + w) = Az + Aw \geq 0 \overset{5.1.10}{\Longrightarrow} z + w \geq 0 \Rightarrow -w \leq z.$

Aus beiden Ungleichungen folgt dann

$$|w| \leq z.$$

□

5.2 Upwind-Verfahren

Bemerkung 5.2.1 (Motivation) [10, p. 45-47]

Wir betrachten nun, ob die Matrix

$$A = \begin{pmatrix} 1 & 0 & \cdots & & & & \\ g_1 & h_1 & j_1 & \cdots & & & \\ 0 & g_2 & h_2 & j_2 & \cdots & & \\ 0 & 0 & g_3 & \ddots & \ddots & & \\ \vdots & \vdots & 0 & \ddots & & & \\ & & & & g_{N-1} & h_{N-1} & j_{N-1} \\ & & & & & 0 & 1 \end{pmatrix}$$

aus (3.7) das M-Matrix-Kriterium erfüllt, wobei

$$\begin{aligned} g_i &= -1 - \frac{h}{2} b(x_i), \\ h_i &= 2 + h^2 \sigma(x_i), \\ j_i &= -1 + \frac{h}{2} b(x_i) \end{aligned}$$

gilt. Dazu prüfen wir die Bedingung $A_{ij} \leq 0$ für alle $i \neq j$.
Es müssen also die beiden Nebendiagonalen mit Einträgen ungleich 0 geprüft werden:

- Nebendiagonale $g_i = -1 - \frac{h}{2} b(x_i) \leq 0$ für alle i, wenn $h > 0$ und $b(x_i) \geq 0$, was für die Standard-Konvektions-Diffusionsgleichung oft gilt. $\rightarrow$ Die linke Nebendiagonale erfüllt das M-Matrix-Kriterium.
- Nebendiagonale $j_i = -1 + \frac{h}{2} b(x_i) \geq 0$, falls $\frac{h}{2} b(x_i) > 1$. Die rechte Nebendiagonale könnte also positiv sein $\rightarrow$ Problem für das M-Matrix-Kriterium.

Dieses positive Vorzeichen auf der rechten Nebendiagonale stammt aus der zentralen Differenz

$$u'(x) \approx \frac{u(x+h) - u(x-h)}{2h}.$$

Um dieses Problem zu lösen, führen wir das Upwind-Verfahren ein.

Bemerkung 5.2.2 [4, p. 26]
Wir betrachten ab jetzt Finite-Differenzen-Verfahren für das Randwertproblem

$$Lu := -\varepsilon u'' + b(x)u' + \sigma(x)u = f(x) \quad \text{für } x \in (0,1), \tag{5.3}$$

mit den Randbedingungen

$$u(0) = u(1) = 0, \tag{5.4}$$

unter den Voraussetzungen

$$\varepsilon > 0,$$

$$b(x) \neq 0 \text{ für alle } x \in [0,1],$$

$$\sigma(x) \geq 0 \text{ für alle } x \in [0,1],$$

wobei die Funktionen $b(x)$, $\sigma(x)$ und $f(x)$ hinreichend glatt seien.
Gemäß [10, p. 47] wollen wir nun statt der zentralen Differenz die einseitige Differenz nutzen:

$$u'(x_i) \approx \frac{u_i - u_{i-1}}{h} \quad \Rightarrow \quad A_{i,i+1} = -\frac{\varepsilon}{h^2} \leq 0.$$

Damit bleiben alle Nebendiagonal-Einträge nichtpositiv, was mit dem M-Matrix-Kriterium verträglich ist.

Definition 5.2.3 (Einfaches Upwind-Verfahren) [4, p. 28]
Das einfache Upwind-Verfahren für das singulär gestörte Randwertproblem (5.3), (5.4) besitzt die Form

$$-\varepsilon \partial^+ \partial^- u_i + b_i \partial^N u_i + \sigma_i u_i = f_i, \quad i = 1, \ldots, N-1, \tag{5.5}$$

mit Randbedingungen

$$u_0 = u_N = 0,$$

wobei

$$\partial^N := \begin{cases} \partial^+, & \text{falls } b < 0, \\ \partial^-, & \text{falls } b > 0. \end{cases}$$

Lemma 5.2.4 [9, p. 48]
Die Koeffizientenmatrix L_h des einfachen Upwind-Schemas ist eine M-Matrix, und das Schema ist gleichmäßig stabil in Bezug auf den Störparameter. Es gilt

$$\|u_h\|_{\infty,h} \leq C_S \|L_h u_h\|_{\infty,h},$$

wobei die Stabilitätskonstante C_S unabhängig von ε und h ist.

Beweis. Der Beweis folgt [9, p. 49]. □

Beispiel 5.2.5
Wir vergleichen nun die exakte Lösung mit der Lösung des zentralen Differenzenschemas und der Lösung des Upwind-Verfahrens. Dazu betrachten wir wieder die Differentialgleichung

$$-\epsilon u''(x) + u'(x) = 0 \tag{5.6}$$

für $x \in (0,1)$, mit $u(0) = 0, u(1) = 1$ aus Beispiel 4.1.2 mit bekannter Lösung

$$u(x) = \frac{e^{\frac{1}{\epsilon}x} - 1}{e^{\frac{1}{\epsilon}} - 1}.$$

Wir implementieren nun die exakte Lösung und die beiden Näherungslösungen in Python:

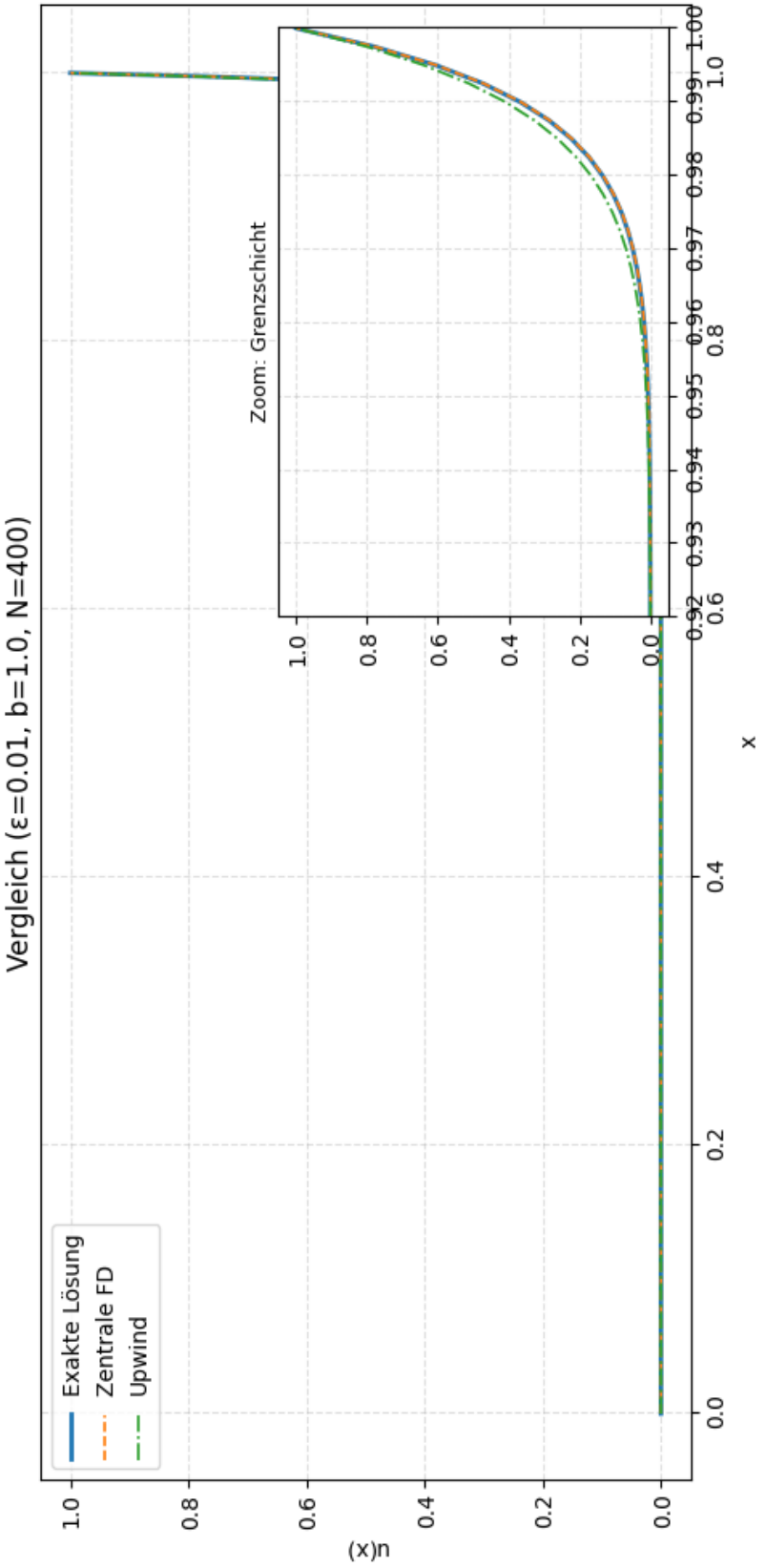

Abbildung 5.1: Vergleich exakte Lösung – zentrales Differenzenverfahren – Upwind-Verfahren

Lemma 5.2.6 (Abschätzung der Norm der Lösung und ihrer Ableitungen) [9, p. 21]
Seien $b(x) \geq \beta > 0$ und $b(x)$, $\sigma(x)$, $f(x)$ hinreichend glatt. Dann erfüllt die Lösung $u(x)$ des Problems (5.3), (5.4)

$$\left|u^{(i)}(x)\right| \leq C\left(1 + \varepsilon^{-i}\exp\left(-\frac{\beta(1-x)}{\varepsilon}\right)\right), \qquad i = 1, 2, \ldots, q,$$

für alle $x \in [0, 1]$. Die maximale Ordnung q hängt von der Glattheit der Daten ab.

Beweis. Der Beweis folgt [9, p. 21] □

Satz 5.2.7 (Konsistenz des einfachen Upwind-Verfahrens) [4, p. 30]
Unter den Voraussetzungen von Bemerkung 5.2.2 mit $b(x) > 0$ und $b(x) \geq \beta > 0$ existiert eine positive Konstante β^*, sodass für den Fehler des einfachen Upwind-Verfahrens (5.5) in den inneren Gitterpunkten $\{x_i : i = 1, \ldots, N-1\}$

$$|u(x_i) - u_i| \leq \begin{cases} C\,h\left(1 + \varepsilon^{-1}\exp\left(-\dfrac{\beta^*(1-x_i)}{\varepsilon}\right)\right), & \text{falls } h < \varepsilon, \\ Ch + C\exp\left(-\dfrac{\beta^*(1-x_{i+1})}{\varepsilon}\right), & \text{falls } h > \varepsilon, \end{cases}$$

gilt.

Korollar 5.2.8 (Konvergenz des einfachen Upwind-Verfahrens außerhalb von Grenzschichten) [4, p. 31]
Unter den Voraussetzungen von Lemma 5.2.4 und Satz 5.2.7 konvergiert das einfache Upwind-Verfahren auf dem Intervall $[0,\, 1-\delta]$ für ein festes $\delta > 0$ von erster Ordnung. Dabei ist die Konvergenzkonstante unabhängig von ε.

Bemerkung 5.2.9 (Fehlerverhalten beim Upwind-Verfahren) [10, p. 53]
Das in Satz 5.2.7 beschriebene Verhalten des einfachen Upwind-Verfahrens kann in numerischen Experimenten zu unerwarteten Ergebnissen führen.
Wird ein äquidistantes Gitter mit $h \gg \varepsilon$ verwendet, liegen alle Knotenpunkte $x_i \in (0, 1)$ außerhalb der Grenzschicht, und die Näherung zeigt eine hohe Genauigkeit. Bei sukzessiver Verfeinerung des Gitters wandern zunehmend Punkte in die Grenzschicht, wo die Näherung nur eine Genauigkeit von Ordnung $\mathcal{O}(1)$ hat. Der maximale Knotenfehler in der diskreten Maximumsnorm kann sich also bei weiterer Gitterverfeinerung vergrößern. Wir betrachten dieses Phänomen in einigen Beispielen genauer.

Beispiel 5.2.10 (Vergleich: Upwind- und Zentraldifferenzenverfahren)

Wir betrachten das Randwertproblem

$$-\varepsilon u''(x) + u'(x) = 0, \qquad x \in (0,1), \quad u(0) = 0,\ u(1) = 1,$$

aus Beispiel 4.1.2 mit der Lösung

$$u(x) = \frac{e^{x/\varepsilon} - 1}{e^{1/\varepsilon} - 1}.$$

Für kleine $\varepsilon > 0$ besitzt u eine starke Grenzschicht bei $x = 1$.
Wir diskretisieren das Problem auf einem äquidistanten Gitter mit N Teilintervallen ($h = 1/N$) mithilfe von

(a) dem **einfachen Upwind-Verfahren**

$$-\varepsilon\frac{u_{i+1}-2u_i+u_{i-1}}{h^2}+\frac{u_i-u_{i-1}}{h}=0,$$

(b) dem **Zentraldifferenzenverfahren**

$$-\varepsilon\frac{u_{i+1}-2u_i+u_{i-1}}{h^2}+\frac{u_{i+1}-u_{i-1}}{2h}=0,$$

unter den Randbedingungen $u_0=0$, $u_N=1$.

Numerische Ergebnisse für $\varepsilon=10^{-3}$
Wir bestimmen den maximalen Fehler

$$E_\infty=\max_i|u(x_i)-u_i|.$$

Die in den folgenden Tabellen angegebenen Werte für h und E_∞ wurden (jeweils in der Mantisse) auf zwei Dezimalstellen gerundet.

(a) Einfaches Upwind-Verfahren:

N	$h=1/N$	E_∞
20	5.00×10^{-2}	1.96×10^{-2}
40	2.50×10^{-2}	3.85×10^{-2}
80	1.25×10^{-2}	7.41×10^{-2}
160	6.25×10^{-3}	1.36×10^{-1}
320	3.13×10^{-3}	1.98×10^{-1}
640	1.56×10^{-3}	1.81×10^{-1}

(b) Zentraldifferenzenverfahren:

N	$h=1/N$	E_∞
20	5.00×10^{-2}	1.41×10^{0}
40	2.50×10^{-2}	8.55×10^{-1}
80	1.25×10^{-2}	7.24×10^{-1}
160	6.25×10^{-3}	5.17×10^{-1}
320	3.13×10^{-3}	2.63×10^{-1}
640	1.56×10^{-3}	8.68×10^{-2}

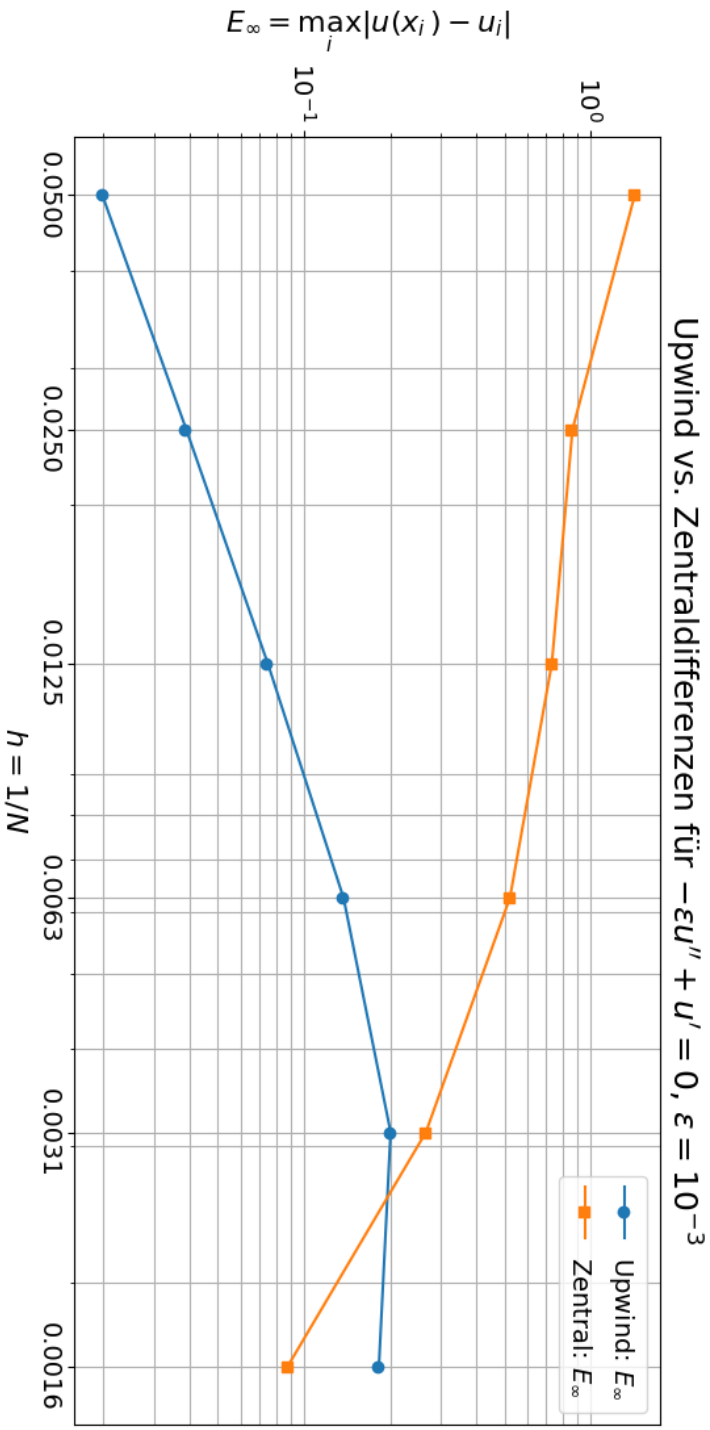

Abbildung 5.2: Fehlerverhalten: Upwind vs. Zentraldifferenzen ($\varepsilon = 10^{-3}$).

Interpretation
Die Ergebnisse zeigen, dass das einfache Upwind-Verfahren numerisch stabil ist. Allerdings

hat es auch bei feinem Gitter einen Fehler in der Größenordnung 10^{-1}. Das Zentraldifferenzenverfahren liefert deutlich größere Fehler.

Bemerkung 5.2.11 (Interpretation des Upwind-Verfahrens als künstliche Diffusion) [4, p. 32-33]
Singulär gestörte Konvektions-Diffusions-Probleme sind numerisch schwierig zu lösen, da die Diffusion ε sich oft um mehrere Größenordnungen von der Konvektion unterscheidet. Das führt zu dünnen Grenzschichten. Das Verhalten des Upwind-Verfahrens lässt sich dabei als Einführung einer künstlichen Diffusion verstehen.

Für $b > 0$ gilt:

$$b_i \partial^N u_i = b_i \frac{u_i - u_{i-1}}{h} = b_i \frac{u_{i+1} - u_{i-1}}{2h} - b_i \frac{u_{i+1} - 2u_i + u_{i-1}}{2h}.$$

Damit kann das einfache Upwind-Verfahren (5.5) in der Form

$$-\left(\varepsilon + \frac{b_i h}{2}\right) \partial^+ \partial^- u_i + b_i \partial^0 u_i + c_i u_i = f_i, \qquad i = 1, \ldots, N-1, \tag{5.7}$$

mit den Randwerten $u_0 = u_N = 0$ geschrieben werden.

Wie in Roos, Stynes und Tobiska [9, p. 52] beschrieben, kann das einfache Upwind-Verfahren als zentrales Differenzenverfahren interpretiert werden, bei dem der Diffusionskoeffizient von ε zu

$$\varepsilon = \varepsilon + \frac{b_i h}{2}$$

geändert wurde. Für $\varepsilon > \frac{b_i h}{2}$ ist die dominante Diffusion weiterhin $\mathcal{O}(\varepsilon)$, während für $\varepsilon < \frac{b_i h}{2}$ die künstliche Diffusion $\mathcal{O}(b_i h)$ überwiegt.

Dieses Verhalten in (5.7) wird als künstliche Diffusion oder künstliche Viskosität bezeichnet. Die Grundidee dieser Methode besteht darin, durch Hinzufügen eines künstlichen Diffusionsterms die numerische Lösung zu stabilisieren.

Eine zu große künstliche Diffusion hat einen Nebeneffekt: die Grenzschichten der Lösung werden „verschmiert“, d.h. sie erscheinen breiter als in der exakten Lösung, da die Schichtbreite direkt vom Diffusionskoeffizienten abhängt
Wir definieren im Folgenden die künstliche Diffusion direkt.

Definition 5.2.12 (Verfahren mit künstlicher Diffusion) [9, p. 52]
Wir definieren ein Finite-Differenzen-Verfahren mit künstlicher Diffusion durch

$$-\varepsilon\, \rho(q(x_i))\, \partial^+ \partial^- u_i + b_i \partial^\circ u_i + \sigma_i u_i = f_i, \qquad i = 1, \ldots, N-1, \tag{5.8}$$

mit den Randbedingungen

$$u_0 = u_N = 0. \tag{5.9}$$

Es gilt

$$q(x) := \frac{b(x)\, h}{2\varepsilon}. \tag{5.10}$$

Dabei wird ρ Fitting-Faktor genannt.

Dieses Verfahren wird auch angepasstes Upwind-Verfahren genannt.

Bemerkung 5.2.13 [9, p. 52]
Für $\rho(q) = 1+q$ erhält man das einfache Upwind-Verfahren. Wir wollen nun untersuchen, welche Wahl von ρ gute Upwind-Verfahren liefert.

Satz 5.2.14 (Stabilität des angepassten Upwind-Verfahrens) [9, p. 52]
Seien

$$b(x) > \beta > 0, \quad \sigma \geq 0, \quad \text{und} \quad \rho(q) > q.$$

Dann gilt:

1. Die Koeffizientenmatrix des angepassten Upwind-Verfahrens 5.8, 5.9, 5.10 ist eine M-Matrix.

2. Das Verfahren ist stabil in der diskreten Maximumsnorm.

3. Die Stabilitätskonstante hängt nicht von ε ab.

Satz 5.2.15 (Konsistenz des angepassten Upwind-Verfahrens) [4, p. 34]
Seien die Voraussetzungen von Satz 5.2.14 erfüllt, sei $u \in C^4([0,1])$ und es gelte

$$|\rho(q) - 1| \leq \min\{q, Mq^2\}$$

für eine Konstante $M > 0$.
Dann ist der Konsistenzfehler des Verfahrens mit künstlicher Diffusion 5.8, 5.9, 5.10 für festes ε von zweiter Ordnung.

Beweis. Der Beweis folgt John [4]. □

5.3 Gleichmäßig konvergente Verfahren

Bemerkung 5.3.1 (Motivation) [4, p. 36]
Wir wollen nun Verfahren entwickeln, die im gesamten Intervall $[0,1]$ gleichmäßig konvergieren, auch innerhalb der Grenzschicht.
Wir betrachten dazu die folgenden beiden Methoden:

- Verfahren durch eine geeignete Wahl der künstlichen Diffusion $\rho(q)$ in (5.8),

- Verfahren durch die Wahl geeigneter Gitter.

Bei sehr kleiner Diffusion müssen numerische Verfahren trotzdem zuverlässig bleiben. Das ist schwierig, weil sich beim Grenzübergang $\varepsilon \to 0$ die Ordnung der Differentialgleichung und damit die Anzahl der nötigen Randbedingungen ändern.

Definition 5.3.2 (Gleichmäßig konvergentes Differenzenschema) [4, p. 36]
Ein Verfahren zur Lösung von (5.3), (5.4) heißt gleichmäßig konvergent von der Ordnung $p > 0$ bezüglich ε in der diskreten Maximumsnorm, wenn es eine Konstante C unabhängig von ε gibt, sodass

$$\|u - u_h\|_{\infty,d} \leq Ch^p$$

gilt.

5.3.1 Geeignete künstliche Diffusion

Bemerkung 5.3.3
Wir wollen nun eine geeignete künstliche Diffusion $\rho(q)$ wählen. Dazu betrachten wir die Lösung von (5.3), (5.4) für $\varepsilon \to 0$. Wir nutzen das reduzierte Problem (vgl. Definition 2.1.6).

Lemma 5.3.4 (Konvergenz gegen reduzierte Lösung) [4, p. 36]
Sei $u(x,\varepsilon)$ die Lösung von (5.3), (5.4), wobei $b(x) \geq \beta > 0$ und $\sigma(x) \geq 0$ gelte. Sei $u_0(x)$ die Lösung des reduzierten Problems.
Dann gilt für alle $x \in [0, x_0)$ mit $x_0 < 1$:

$$\lim_{\varepsilon \to 0} u(x,\varepsilon) = u_0(x).$$

Beweis. Der Beweis folgt John [4]. □

Lemma 5.3.5 [4, p. 37]
Unter den Voraussetzungen von Lemma 5.3.4 existiert eine von x und ε unabhängige Konstante $C > 0$, so dass für die Lösung von (5.3), (5.4) gilt:

$$\left| u(x,\varepsilon) - \Big(u_0(x) - u_0(1) \exp\Big(-\frac{b(1)(1-x)}{\varepsilon} \Big) \Big) \right| \leq C\,\varepsilon, \qquad x \in [0,1].$$

Bemerkung 5.3.6 (Notwendige Bedingung für eine geeignete künstliche Diffusion $\rho(q)$) [4, p. 37]
Seien $\rho^* := h/\varepsilon$ und i fest. Das bedeutet, dass für $h \to 0$ auch $\varepsilon \to 0$ gilt. Ziel ist es, unter diesen Bedingungen eine geeignete Funktion $\rho(q)$ zu bestimmen.
Aus Lemma 5.3.5 folgt für $\varepsilon \to 0$ (wenn $h \to 0$):

$$\begin{aligned}
\lim_{h\to 0} u(1-ih) &= \lim_{h\to 0} \left[u_0(1-ih) - u_0(1-ih) \exp\Big(-\frac{b(1)(1-(1-ih))}{\varepsilon} \Big) \right] \\
&= u_0(1) - u_0(1) \lim_{h\to 0} \exp\Big(-\frac{b(1) ih}{\varepsilon} \Big) \\
&= u_0(1)\big(1 - \exp(-ib(1)\rho^*) \big) \\
&= u_0(1)\big(1 - \exp(-2iq(1)) \big). \qquad (5.11)
\end{aligned}$$

Das angepasste Upwind-Verfahren hat die Form

$$-\varepsilon \rho(q(b_i)) \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + b_i \frac{u_{i+1} - u_{i-1}}{2h} = f_i - \sigma_i u_i,$$

oder nach Erweiterung mit h^2/ε:

$$-\rho(q(b_i))(u_{i+1} - 2u_i + u_{i-1}) + q(b_i)(u_{i+1} - u_{i-1}) = h\rho^*(f_i - \sigma_i u_i).$$

Für den rechten Rand ($i = N-1$) gilt:

$$\lim_{h\to 0} \big(-\rho(q_{N-1})(u_N - 2u_{N-1} + u_{N-2}) + q_{N-1}(u_N - u_{N-2}) \big) = 0.$$

Einsetzen von (5.11) (mit $i \in \{0,1,2\}$) und der Annahme $u_0(1) \neq 0$) ergibt:

$$0 = -\rho(q(1))\big(-\mathrm{e}^0 + 2\mathrm{e}^{-2q(1)} - \mathrm{e}^{-4q(1)} \big) + q(1)\big(-\mathrm{e}^0 + \mathrm{e}^{-4q(1)} \big).$$

Aus

$$\frac{-1+\mathrm{e}^{-4x}}{-1+2\mathrm{e}^{-2x}-\mathrm{e}^{-4x}} = \frac{(\mathrm{e}^{-2x}-1)(\mathrm{e}^{-2x}+1)}{-(\mathrm{e}^{-2x}-1)^2} = \frac{1+\mathrm{e}^{-2x}}{1-\mathrm{e}^{-2x}} = \frac{\mathrm{e}^{x}+\mathrm{e}^{-x}}{\mathrm{e}^{x}-\mathrm{e}^{-x}} = \coth(x)$$

folgt

$$\rho(q(1)) = q(1)\frac{1-\mathrm{e}^{-4q(1)}}{1-2\mathrm{e}^{-2q(1)}+\mathrm{e}^{-4q(1)}} = q(1)\coth(q(1)).$$

Eine passende Wahl, die diesen Grenzwert erfüllt, ist daher

$$\rho(q) = q\ \coth(q).$$

Diese Funktion erfüllt auch die Bedingungen für die Konsistenz von Verfahren mit künstlicher Diffusion gemäß Satz 5.2.15.

Für kleine Werte von q gilt $\rho(q) \approx 1$, sodass in diesem Bereich keine zusätzliche Diffusion eingebracht wird. Für große Werte von q gilt näherungsweise $\rho(q) \approx q$. Dadurch wird der Diffusionsterm im Differenzenschema verstärkt.Auf diese Weise wird die künstliche Diffusion nur dort eingesetzt, wo sie für die numerische Stabilität erforderlich ist.
Die folgende Abbildung zeigt das Verhalten der künstlichen Diffusion $\rho(q) = q\ \coth(q)$ im Vergleich zum Upwind-Verfahren mit $\rho(q) = 1 + q$.

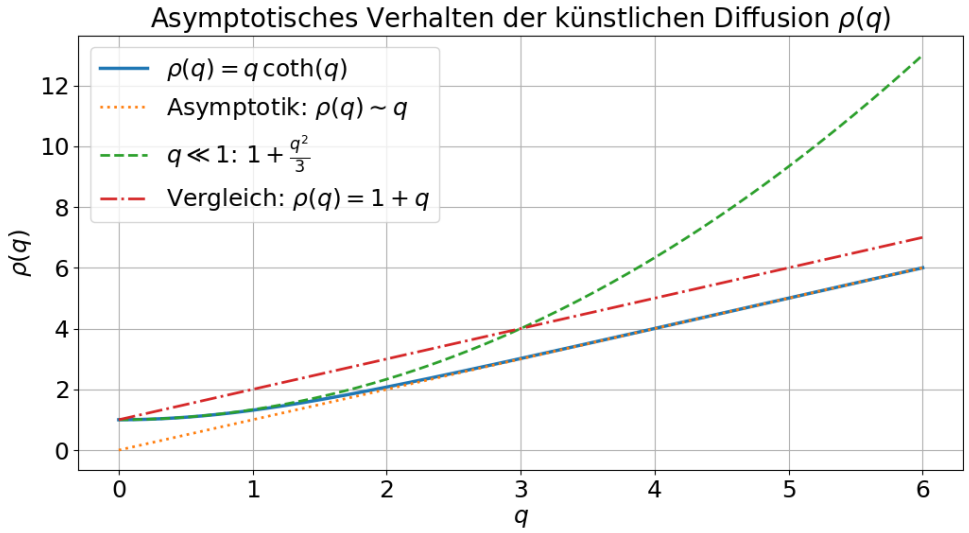

Abbildung 5.3: Künstliche Diffusion

Theorem 5.3.7 (Zwei notwendige Bedingungen für gleichmäßige Konvergenz auf einem äquidistanten Gitter) [10, p. 56]
Angenommen, wir haben ein äquidistantes Gitter mit Schrittweite $h = 1/N$ für eine positive ganze Zahl N. Wir nehmen an, dass ein Differenzenschema für das Problem

$$-\varepsilon u'' + au = f, \quad u(0) = g_0, \quad u(1) = g_1, \text{ mit positiver Konstante } a,$$

in folgender Form geschrieben werden kann:

$$\theta_- u_{i-1} + \theta_0 u_i + \theta_+ u_{i+1} = hf_i, \quad i = 1, \ldots, N-1, \tag{5.12a}$$

$$u_0 = g_0, \quad u_N = g_1. \tag{5.12b}$$

Dabei hängt jedes $\theta = \theta(h, \varepsilon)$ nur vom Verhältnis h/ε ab. Falls das Schema für ein $\beta > 0$ gleichmäßig konvergent ist, so gilt:

$$\theta_- + \theta_0 + \theta_+ = 0 \quad \text{und} \quad e^{-ah/\varepsilon}\theta_- + \theta_0 + e^{ah/\varepsilon}\theta_+ = 0. \tag{5.13}$$

Beweis. Der Beweis folgt Stynes und Stynes [10, p. 56-57]. □

Beispiel 5.3.8 (Das Il'in–Allen–Southwell-Differenzschema) [4, p. 38]
Das Verfahren

$$-\frac{h}{2}b_i \coth\left(\frac{\mu h}{2\varepsilon}b_i\right))\partial^+\partial^- u_i \;+\; b_i\partial^\circ u_i \;+\; \sigma_i u_i \;=\; f_i, \qquad i = 1, \ldots, N-1,$$

mit Randbedingungen

$$u_0 = u_N = 0$$

wird Il'in-Verfahren bzw. Il'in-Allen-Southwell-Verfahren genannt.

Theorem 5.3.9 (Uniforme Konvergenz des Il'in-Allen-Southwell-Schemas) [9, p. 60]
Sei $b(x) > \beta > 0$. Dann ist das Il'in-Allen-Southwell-Schema gleichmäßig konvergent von erster Ordnung in der diskreten Maximums-Norm:

$$\|u - u_h\|_{\infty,d} \leq Ch.$$

Beweis. Der Beweis folgt Theorem 11.34 aus [1, p. 535]. □

Bemerkung 5.3.10 [9, p. 61]
Falls $\sigma(x) \equiv 0$ und $b(x)$ sowie $f(x)$ konstant sind, ist das Il'in-Allen-Southwell-Schema exakt, d.h. $u_i = u(x_i)$ für alle i.

Bemerkung 5.3.11 [9, p. 60]
Eine Untersuchung des Verhaltens des Il'in-Allen-Southwell-Schemas bei Anwendung auf das Beispiel

$$-\varepsilon u'' + u' = x, \quad u(0) = u(1) = 0,$$

zeigt, dass außerhalb der Randschicht die Ordnung der gleichmäßigen Konvergenz nur eins ist.

Beispiel 5.3.12 (Vergleich von Upwind-, Zentral- und Il'in-Allen-Southwell-Verfahren)

Dieses Beispiel wird in Anlehnung an [4, p. 39] betrachtet. Gegeben ist das Randwertproblem

$$-\varepsilon u''(x) + u'(x) = 1 \quad \text{auf } (0,1), \qquad u(0) = u(1) = 0,$$

mit $\varepsilon = 10^{-3}$. Wir betrachten nun wieder den Fehler

$$E_\infty = \max_i |u(x_i) - u_i|.$$

N	$h = 1/N$	Einf. Upwind	Zentral	IAS
2	0.50000	0.00199	124.50000	0.0
4	0.25000	0.00398	31.00400	0.0
8	0.12500	0.00794	7.71502	0.0
16	0.06250	0.01575	2.02352	0.0
20	0.05000	0.01961	1.40904	1.11×10^{-16}
40	0.02500	0.03846	0.85489	4.11×10^{-15}
80	0.01250	0.07407	0.72414	1.11×10^{-16}
160	0.00625	0.13600	0.51708	2.22×10^{-16}
320	0.00313	0.19849	0.26345	2.89×10^{-15}
640	0.00156	0.18063	0.08680	5.20×10^{-14}

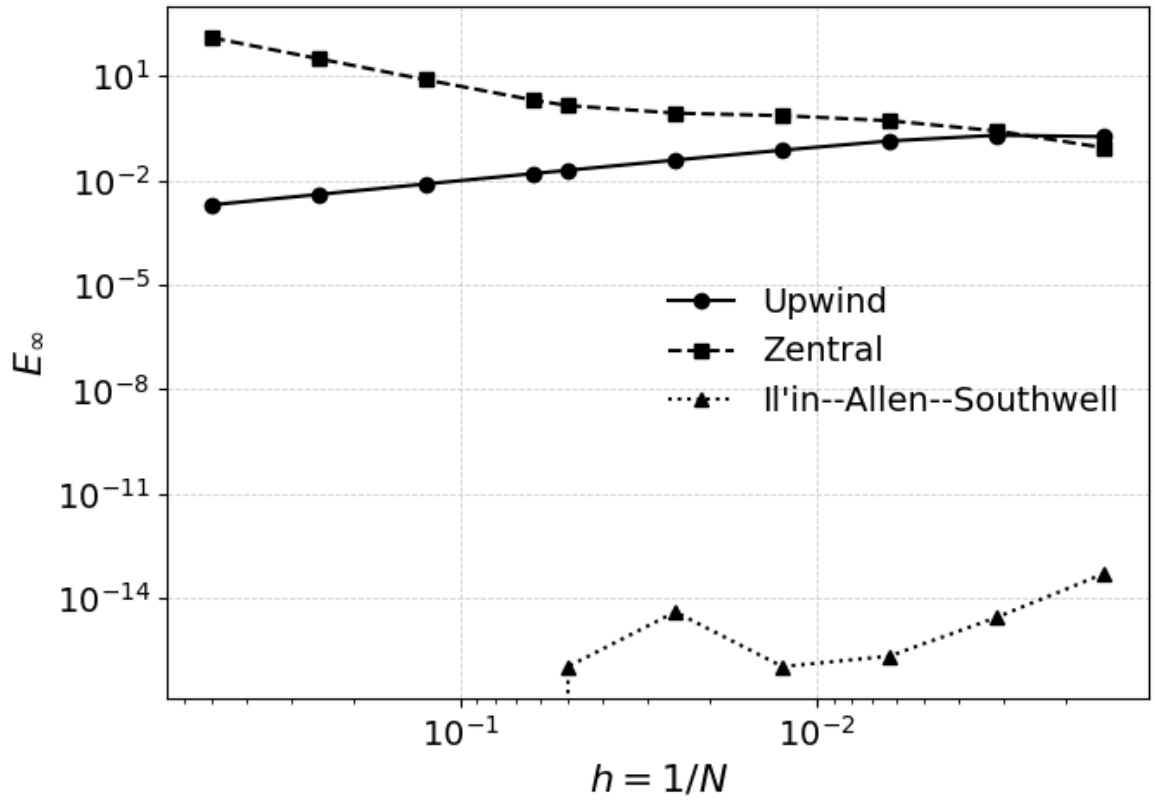

Abbildung 5.4: Fehler E_∞ für $\varepsilon = 10^{-3}$.

Man kann erkennen, dass das Il'in–Allen–Southwell-Verfahren die geringsten Fehler aufweist und für große Schrittweiten exakte Ergebnisse in den Knoten liefert.

5.3.2 Shishkin-Gitter

Bemerkung 5.3.13 [10, p. 60]
Zur numerischen Lösung von Konvektions–Diffusionsproblemen kann anstelle eines äquidistanten Gitters ein Gitter verwendet werden, das die Gitterpunkte in der Grenzschicht bei $x = 1$

verdichtet. Wir betrachten das Problem (2.1) und setzen

$$\tau = \min\left\{\tfrac{1}{2},\ \frac{2}{\alpha}\varepsilon \ln N\right\}.$$

Da der Fall $\tau = \frac{1}{2}$ nur auftritt, wenn N im Verhältnis zu ε exponentiell groß ist und daher in der Praxis kaum vorkommt, setzt man gewöhnlich

$$\tau = \frac{2}{\alpha}\varepsilon \ln N.$$

Der Übergangspunkt des Shishkin-Gitters, welcher den groben vom feinen Gitterbereich trennt, ist somit $1-\tau$ und liegt typischerweise nahe bei 1. Für eine gerade Zahl N unterteilt man sowohl das Intervall $[0, 1-\tau]$ als auch das Intervall $[1-\tau, 1]$ jeweils in $N/2$ äquidistante Teilintervalle.
Der grobe Teil des Shishkin-Gitters hat Schrittweite

$$H := \frac{2(1-\tau)}{N},$$

woraus aufgrund $0 \le \tau \le \frac{1}{2}$ folgt, dass

$$N^{-1} \ \le\ H \ \le\ 2N^{-1}.$$

Im feinen Teil ist die Schrittweite

$$h := \frac{2\tau}{N} = \frac{4}{\alpha}\,\varepsilon\,\frac{\ln N}{N},$$

so dass $h \ll \varepsilon$ gilt. Die Gitterpunkte sind damit

$$x_i = iH, \qquad i = 0, \ldots, \tfrac{N}{2}$$

für den groben Bereich, sowie

$$x_i = 1 - (N-i)\,h, \qquad i = \tfrac{N}{2}+1, \ldots, N$$

für den feinen Bereich. Wir setzen für alle i

$$h_i = x_i - x_{i-1}.$$

Satz 5.3.14 (Shishkin-Zerlegung von u) [10, p. 40]
Sei q eine positive ganze Zahl und sei u die Lösung von (2.1). Wir nehmen an, dass die Funktionen σ, b und f hinreichend glatt sind. Dann existiert eine Zerlegung

$$u = S + E,$$

sodass

$$\|S^{(j)}\|_\infty \le C, \tag{5.14}$$
$$|E^{(j)}(x)| \le C\,\varepsilon^{-j} e^{-\alpha(1-x)/\varepsilon}, \quad 0 \le x \le 1, \tag{5.15}$$

für $0 \le j \le q$, wobei die Konstante $C = C(q)$ ist und $\alpha > 0$ eine von ε unabhängige Konstante darstellt. Zusätzlich gilt

$$LS(x) = f(x) \quad \text{und} \quad LE(x) = 0 \quad \text{für } 0 \le x \le 1.$$

Beweis. Der Beweis folgt Stynes und Stynes [10]. □

Bemerkung 5.3.15 (Upwind-Verfahren auf dem Shishkin-Gitter) [10, p. 61]

Wir analysieren nun das einfache Upwind-Verfahren auf einem Shishkin-Gitter.
Für jede Gitterfunktion $v_h = (v_0, \ldots, v_N)^\top$ definieren wir:

$$\partial^- v_i = \frac{v_i - v_{i-1}}{h_i}, \quad \delta^2 v_i = \frac{2}{h_i + h_{i+1}} \left(\partial^- v_{i+1} - \partial^- v_i \right).$$

Das ist eine Standard-Diskretisierung von $v_i''(x_i)$ auf einem nicht-äquidistanten Gitter.
Unser einfaches Upwind-Differenzenschema lautet:

$$-\varepsilon\, \delta^2 u_i + a_i \partial^- u_i + b_i u_i = f_i, \quad i = 1, \ldots, N-1, \tag{5.16}$$

$$u_0 = u_N = 0. \tag{5.17}$$

Die zugehörige Matrix L_N ist eine M-Matrix.
Wir zerlegen die exakte Lösung u nach Satz 5.3.14:

$$u = S + E.$$

Analog zerlegen wir die diskrete Lösung u_i in

$$u_i = S_i + E_i,$$

wobei

$$L_N S_i = (LS)_i, \quad i = 1, \ldots, N-1, \quad S_0 = S(0), \quad S_N = S(1), \tag{5.18}$$

$$L_N E_i = (LE)_i = 0, \quad i = 1, \ldots, N-1, \quad E_0 = E(0), \quad E_N = E(1). \tag{5.19}$$

Der Gesamtfehler lässt sich aufteilen:

$$|u(x_i) - u_i| = |(S+E)(x_i) - (S_i + E_i)| \leq |S(x_i) - S_i| + |E(x_i) - E_i|.$$

Lemma 5.3.16 [10, p. 61]
Es existiert eine Konstante C_0 so, dass

$$|S(x_i) - S_i| \leq C_0\, N^{-1} \quad \text{für } i = 0, \ldots, N.$$

Beweis. Der Beweis folgt Stynes und Stynes [10]. □

Lemma 5.3.17 [10, p. 63]
Es existiert eine Konstante C derart, dass

$$|E_i| \leq C\, N^{-1} \quad \text{für } i = 0, \ldots, \frac{N}{2}.$$

Beweis. Der Beweis folgt Stynes und Stynes [10]. □

Korollar 5.3.18 [10, p. 63]
Es existiert eine Konstante C so, dass

$$|E(x_i) - E_i| \leq C\, N^{-1} \quad \text{für } i = 0, \ldots, \frac{N}{2}.$$

Beweis. Der Beweis folgt Stynes und Stynes [10]. □

Lemma 5.3.19 [10, p. 64]
Es existiert eine Konstante C so, dass

$$|E(x_i) - E_i| \leq C\, N^{-1} \ln N \quad \text{für } i = \frac{N}{2} + 1, \ldots, N.$$

Beweis. Der Beweis folgt Stynes und Stynes [10]. □

Theorem 5.3.20 (Gleichmäßige Konvergenz des einfachen Upwind-Verfahrens auf einem Shishkin-Gitter) [10, p. 64]
Es existiert eine Konstante C, so dass für die Lösung des einfachen Upwind-Verfahrens auf einem Shishkin-Gitter gilt:

$$\|u(x_i) - u_i\|_{\infty,d} \leq C\, N^{-1} \ln N \quad \text{für } i = 0, \ldots, N.$$

Beweis. Der Beweis folgt Stynes und Stynes [10]. □

Bemerkung 5.3.21 [10, p. 65]
Die genaue Wahl des Übergangspunktes $1-\tau$ im Shishkin-Gitter ist aus theoretischer und aus numerischer Sicht von Bedeutung. Eine Betrachtung des Beweises von Theorem 5.3.20 zeigt, dass τ die Form

$$\tau = \frac{k}{\alpha}\, \varepsilon\, \phi(N)$$

haben sollte, wobei $\phi(N) \to \infty$ und $N^{-1}\phi(N) \to 0$ für $N \to \infty$ gilt, und k eine Konstante ist. Die einfachste Wahl für $\phi(N)$ ist $\ln N$.

Bemerkung 5.3.22 [10, p. 65]
Für das Zentrale-Differenzen-Verfahren auf einem Shishkin-Gitter genügt die berechnete Lösung u_i der Abschätzung

$$|u(x_i) - u_i| \leq CN^{-2}(\ln N)^2 \quad \text{für alle } i.$$

Der Beweis erfordert Geschicklichkeit, da die zugehörige Matrix keine M-Matrix ist und das Verfahren kein diskretes Maximumprinzip erfüllt.

Beispiel 5.3.23 (Numerisches Beispiel zum einfachen Upwind-Verfahren auf einem Shishkin-Gitter)
Wir betrachten das Randwertproblem

$$-\varepsilon u''(x) + u'(x) = 0, \qquad x \in (0,1), \tag{5.20}$$

mit den Randbedingungen

$$u(0) = 0, \qquad u(1) = 1. \tag{5.21}$$

Für $\varepsilon = 10^{-2}$ ist die exakte Lösung gegeben durch

$$u(x) = \frac{e^{x/\varepsilon} - 1}{e^{1/\varepsilon} - 1}, \tag{5.22}$$

und besitzt bei $x = 1$ eine Grenzschicht.

Zur numerischen Lösung von (5.20)–(5.21) nutzen wir das einfache Upwind-Verfahren. Dabei werden zwei unterschiedliche Gitter betrachtet:

- ein äquidistantes Gitter mit Schrittweite $h = 1/N$,
- ein Shishkin-Gitter mit

$$\tau = \min\left\{\frac{1}{2}, 2\varepsilon \ln N\right\}$$

und Übergangspunkt

$$x = 1 - \tau.$$

Die zugehörigen Schrittweiten sind

$$H = \frac{2(1-\tau)}{N} \quad \text{und} \quad h = \frac{2\tau}{N}.$$

Der Fehler

$$E_\infty = \max_i |u(x_i) - u_i|. \tag{5.23}$$

wird in der diskreten Maximmumsnorm für verschiedene Werte von N berechnet.

Zusätzlich wird die numerische Lösung in einem vergrößerten Ausschnitt nahe der Grenzschicht bei $x = 1$ dargestellt.

N	τ	E_∞ (uniform)	E_∞ (Shishkin)
8	$4.16 \cdot 10^{-2}$	$7.41 \cdot 10^{-2}$	$1.61 \cdot 10^{-1}$
16	$5.55 \cdot 10^{-2}$	$1.36 \cdot 10^{-1}$	$1.06 \cdot 10^{-1}$
32	$6.93 \cdot 10^{-2}$	$1.98 \cdot 10^{-1}$	$6.84 \cdot 10^{-2}$
64	$8.32 \cdot 10^{-2}$	$1.81 \cdot 10^{-1}$	$4.35 \cdot 10^{-2}$
128	$9.70 \cdot 10^{-2}$	$1.06 \cdot 10^{-1}$	$2.63 \cdot 10^{-2}$
256	$1.11 \cdot 10^{-1}$	$6.21 \cdot 10^{-2}$	$1.54 \cdot 10^{-2}$
512	$1.25 \cdot 10^{-1}$	$3.32 \cdot 10^{-2}$	$8.79 \cdot 10^{-3}$
1024	$1.39 \cdot 10^{-1}$	$1.73 \cdot 10^{-2}$	$4.93 \cdot 10^{-3}$

Tabelle 5.1: Maximaler Fehler E_∞ für das Upwind-Verfahren auf äquidistantem und Shishkin-Gitter ($\varepsilon = 10^{-2}$).

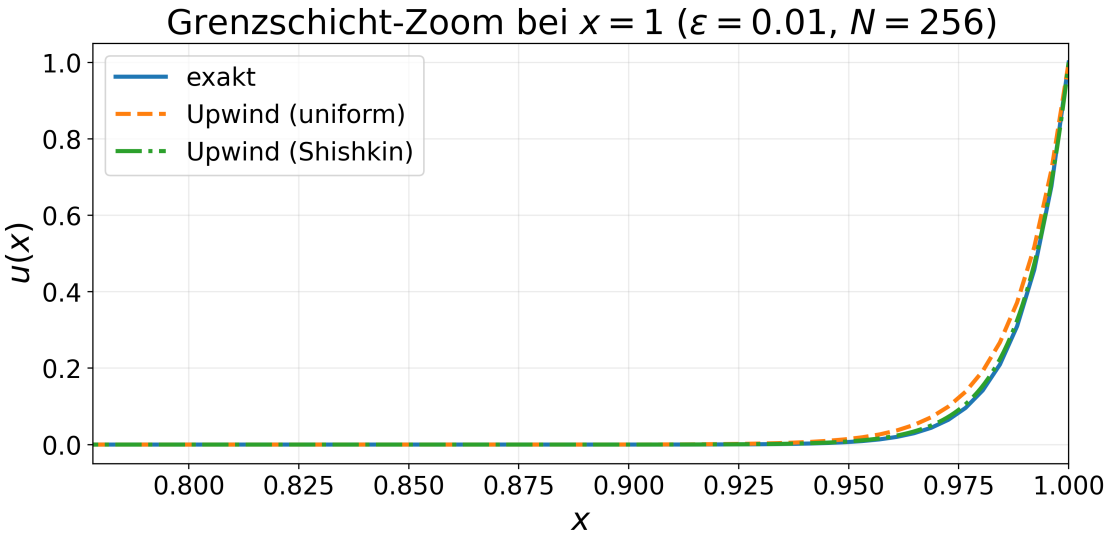

Abbildung 5.5: Zoom der Lösung in die Grenzschicht bei $x = 1$ für $\varepsilon = 10^{-2}$ und $N = 256$.

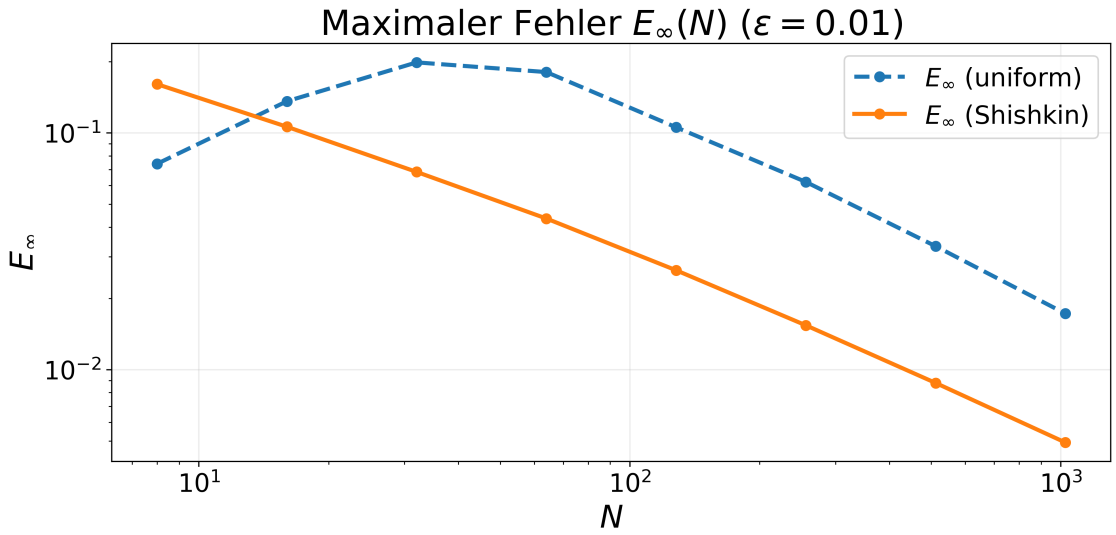

Abbildung 5.6: Zoom der Lösung in die Grenzschicht bei $x = 1$ für $\varepsilon = 10^{-2}$ und $N = 256$.

Man kann erkennen, dass der maximale Fehler E_∞ auf dem Shishkin-Gitter für wachsendes N deutlich geringer ist als auf dem äquidistanten Gitter.

Kapitel 6

Ausblick

Die Ergebnisse in dieser Arbeit zeigen die Schwierigkeiten der numerischen Lösung von konvektionsdominanten Randwertproblemen auf. Für kleine Diffusionsparameter ε treten ausgeprägte Grenzschichten auf, in denen sich die Lösung auf sehr kurzen Intervallen stark ändert, während sie außerhalb dieser Bereiche glatt verläuft. Dadurch erhalten wir mit Zentralen-Differenzen-Verfahren auf äquidistanten Gittern häufig keine zufriedenstellenden Resultate.
Das Upwind-Verfahren, das eine zugehörige M-Matrix besitzt, erweist sich als stabil, liefert aber zu ungenaue Werte, da es eine Konsistenzordnung von 1 hat. Die Grenzschichten werden verschmiert.
Weitere Untersuchungen haben gezeigt, dass durch künstliche Diffusion (hier durch das Il'in-Allen-Southwell-Schema) sehr genaue Werte erzielt werden können.
Shishkin-Gitter bieten hierfür eine andere Möglichkeit und erlauben es, eine gleichmäßige Konvergenz zu erhalten.
Als Weiterführung bieten sich Untersuchungen zu Verfahren höherer Ordnung an. Hier kann man betrachten, inwiefern sich die hier vorgestellten Konzepte auf höherdimensionale Probleme übertragen lassen.
Darüber hinaus stellt die Finite-Elemente-Methode (FEM) einen weiteren Ansatz zur Lösung konvektionsdominanter Randwertprobleme dar. Durch geeignete Verfahren können numerische Oszillationen reduziert und Grenzschichten zuverlässiger abgebildet werden.
Insgesamt verdeutlicht diese Arbeit, dass eine Kombination des analytischen Verhaltens der Lösung, stabiler Diskretisierung und geeigneter Gitter- und Diffusionswahl wesentlich für eine zuverlässige numerische Lösung konvektionsdominanter Probleme ist.

Literatur

[1] Gabriel R. Barrenechea, Volker John und Petr Knobloch. *Monotone discretizations for elliptic second order partial differential equations*. Bd. 61. Springer Series in Computational Mathematics. Springer, Cham, [2025] ©2025, S. xii+649. ISBN: 978-3-031-80683-4; 978-3-031-80684-1. DOI: 10.1007/978-3-031-80684-1. URL: https://doi.org/10.1007/978-3-031-80684-1.

[2] Mona Berciu. *Definitions and Important Facts Regarding ODEs and PDEs*. Lecture notes, Department of Physics and Astronomy, University of British Columbia. Section: Boundary Conditions. 2006. URL: https://www.phas.ubc.ca/~berciu/TEACHING/PHYS312/LECTURES/FILES/defs.pdf.

[3] Etienne Emmrich. *Gewöhnliche und Operator-Differentialgleichungen*. Eine integrierte Einführung in Randwertprobleme und Evolutionsgleichungen für Studierende. Wiesbaden: Friedr. Vieweg & Sohn Verlag, 2004. ISBN: 3-528-03213-8.

[4] Volker John. *Numerik konvektions-dominanter Probleme*. Vorlesungsskript, Universität des Saarlandes, Saarbrücken. 2010.

[5] Volker John. *Numerik partieller Differentialgleichungen: Eine elementare Einführung*. Vorlesungsskript, Universität des Saarlandes, Saarbrücken. 2009.

[6] Anne Kværnø und Andreas Massing. *Finite Difference Methods for Two-Point Boundary Value Problems*. Lecture Notes, Norwegian University of Science and Technology (NTNU). 2021.

[7] Randall J. LeVeque. *Finite difference methods for ordinary and partial differential equations*. Steady-state and time-dependent problems. Society for Industrial und Applied Mathematics (SIAM), Philadelphia, PA, 2007, S. xvi+341. ISBN: 978-0-898716-29-0. DOI: 10.1137/1.9780898717839. URL: https://doi.org/10.1137/1.9780898717839.

[8] Zhilin Li, Zhonghua Qiao und Tao Tang. *Numerical solution of differential equations*. Introduction to finite difference and finite element methods. Cambridge University Press, Cambridge, 2018, S. ix + 293. ISBN: 978-1-316-61510-2; 978-1-107-16322-5.

[9] Hans-Görg Roos, Martin Stynes und Lutz Tobiska. *Robust numerical methods for singularly perturbed differential equations*. Second. Bd. 24. Springer Series in Computational Mathematics. Convection-diffusion-reaction and flow problems. Springer-Verlag, Berlin, 2008, S. xiv+604. ISBN: 978-3-540-34466-7.

[10] Martin Stynes und David Stynes. *Convection-diffusion problems*. Bd. 196. Graduate Studies in Mathematics. An introduction to their analysis and numerical solution. American Mathematical Society, Providence, RI; Atlantic Association for Research in the Mathematical Sciences (AARMS), Halifax, NS, 2018, S. viii+156. ISBN: 978-1-4704-4868-4.

Anhang A

Quellcode (Python)

Python-Code zur Erzeugung von Abbildungen 2.1 und 2.2

```python
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.size'] = 18
plt.rcParams['axes.titlesize'] = 18
plt.rcParams['axes.labelsize'] = 18
plt.rcParams['xtick.labelsize'] = 16
plt.rcParams['ytick.labelsize'] = 16
plt.rcParams['legend.fontsize'] = 16

def u_exact(x, eps): return x - (np.exp(-(1-x)/eps) - np.exp(-1/eps)) / (1 - np.exp(-1/eps))

def u_asym(x, eps): return x - np.exp(-(1-x)/eps)

x = np.linspace(0, 1, 1200)
eps_list = [0.2, 0.1, 0.05]
plt.figure(figsize=(9, 5.5))
for eps in eps_list:
    plt.plot(x, u_exact(x, eps), label=f"exakt, ={eps:g}")
    plt.plot(x, u_asym(x, eps), linestyle="--", label=f"asympt., ={eps:g}")

plt.xlabel("x")
plt.ylabel("u(x)")
plt.title("Exakte vs. asymptotische Lösung")
plt.legend()
plt.tight_layout()
plt.show()

plt.figure(figsize=(9, 5.5))
for eps in eps_list:
    error = np.abs(u_exact(x, eps) - u_asym(x, eps))
    plt.semilogy(x, error, label=f"={eps:g}")
plt.xlabel("x")
plt.ylabel(r"$|u_{\mathrm{exakt}} - u_{\mathrm{as}}|$")
```

```
plt.title("Fehler der asymptotischen Lösung")
plt.legend()
plt.tight_layout()
plt.show()
```

Python-Code zur Erzeugung von Abbildungen 3.2, 3.3, 3.4 und Tabelle 3.1

```
import numpy as np
from scipy.linalg import solve
import matplotlib.pyplot as plt

plt.rcParams.update({
    "font.size": 14,
    "axes.labelsize": 14,
    "xtick.labelsize": 12,
    "ytick.labelsize": 12,
    "legend.fontsize": 12,
    "figure.figsize": (7, 4),
    "savefig.dpi": 300
})

def b_func(x):
    return 0.0

def sigma_func(x):
    return 0.0

def f_func(x):
    #-u'' = f
    return np.pi**2 * np.sin(np.pi * x)

def u_exact(x):
    return np.sin(np.pi * x)

N_values = [4, 8, 16]

for N in N_values:
    h = 1.0 / N
    x = np.linspace(0.0, 1.0, N+1) #Gitterpunkte


    A = np.zeros((N+1, N+1))
    rhs = np.zeros(N+1)

    #Randbedingungen
    A[0, 0] = 1.0
    A[N, N] = 1.0
    rhs[0] = 0.0
    rhs[N] = 0.0
```

```
43
44      for i in range(1, N):
45          bi = b_func(x[i])
46          sigma_i = sigma_func(x[i])
47
48          g_i = -1.0 - 0.5 * h * bi
49          h_i =  2.0 + (h**2) * sigma_i
50          j_i = -1.0 + 0.5 * h * bi
51
52          A[i, i-1] = g_i
53          A[i, i]   = h_i
54          A[i, i+1] = j_i
55
56          rhs[i] = (h**2) * f_func(x[i])
57
58      U = solve(A, rhs)
59      U_exact = u_exact(x)
60
61      error = U - U_exact
62      error_L2  = np.sqrt(np.sum(error**2) * h) #L2-Norm
63      error_max = np.max(np.abs(error)) #Diskrete Maximumsnorm
64
65      print(f"N={N}, L2-Fehler={error_L2:.3e}, Max-Fehler={error_max:.3e}")
66
67      plt.figure()
68      plt.plot(x, U, 'o-', label='Numerisch', markersize=6)
69      plt.plot(x, U_exact, '--', label='Exakt')
70
71      plt.xlabel(r"$x$")
72      plt.ylabel(r"$u(x)$")
73      plt.title(f"Numerische vs. exakte Lösung, N={N}")
74      plt.legend()
75      plt.grid(True)
76
77      plt.savefig(f"FDM_N_{N}.png", bbox_inches='tight')
78      plt.show()
79
```

Python-Code zur Erzeugung von Abbildungen 4.1 und 4.2

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import matplotlib as mpl
4
5  mpl.rcParams.update({
6      "figure.figsize": (6, 4),
7      "figure.dpi": 300,
8      "font.size": 11,
9      "axes.titlesize": 11,
```

```
    "axes.labelsize": 11,
    "xtick.labelsize": 10,
    "ytick.labelsize": 10,
    "legend.fontsize": 10,
    "lines.linewidth": 1.2,
    "axes.linewidth": 0.8,
    "grid.alpha": 0.25,
    "grid.linestyle": ":",
})

x = np.linspace(0, 1, 600)
epsilon = [1, 0.1, 0.01]
linestyles = ["-", "--", "-."]
color = "black"

fig1, ax1 = plt.subplots()

for eps, ls in zip(epsilon, linestyles):
    u = (np.exp(x/eps) - 1) / (np.exp(1/eps) - 1)
    ax1.plot(x, u, linestyle=ls, color=color, label=fr"$\varepsilon = {eps}$")

ax1.set_title(r"Lösung $u(x)$ für verschiedene $\varepsilon$")
ax1.set_xlabel(r"$x$")
ax1.set_ylabel(r"$u(x)$")

ax1.yaxis.grid(True)
ax1.xaxis.grid(False)
ax1.legend(loc="best", frameon=False)

fig1.tight_layout()
fig1.savefig(save_path + r"\loesung_gesamt.pdf", bbox_inches="tight")
plt.close(fig1)

fig2, ax2 = plt.subplots()

for eps, ls in zip(epsilon, linestyles):
    u = (np.exp(x/eps) - 1) / (np.exp(1/eps) - 1)
    ax2.plot(x, u, linestyle=ls, color=color, label=fr"$\varepsilon = {eps}$")


ax2.set_xlim(0.9, 1.0)
ax2.set_ylim(0, 1.05)

ax2.set_title(r"Grenzschicht bei $x = 1$")
ax2.set_xlabel(r"$x$")
ax2.set_ylabel(r"$u(x)$")

ax2.yaxis.grid(True)
ax2.xaxis.grid(False)
ax2.legend(loc="best", frameon=False)

```

```
fig2.tight_layout()
fig2.savefig(save_path + r"\grenzschicht_rechts.pdf", bbox_inches="tight")
plt.close(fig2)
```

Python-Code zur Erzeugung von Abbildung 5.1

```
import numpy as np
import matplotlib.pyplot as plt

def exact_solution(x, eps, b):
    with np.errstate(over='ignore', invalid='ignore'):
        num = np.exp(b * x / eps) - 1.0
        den = np.exp(b / eps) - 1.0
        return num / den

def build_central_matrix(eps, b, h, M):
    A = np.zeros((M, M), dtype=float)
    for i in range(M):
        A[i,i] = 2*eps / h**2
        if i > 0:
            A[i,i-1] = -eps / h**2 - b / (2*h)
        if i < M-1:
            A[i,i+1] = -eps / h**2 + b / (2*h)
    return A

def build_upwind_matrix(eps, b, h, M):
    A = np.zeros((M, M), dtype=float)
    for i in range(M):
        A[i,i] = (2*eps + b*h) / h**2
        if i > 0:
            A[i,i-1] = -(eps + b*h) / h**2
        if i < M-1:
            A[i,i+1] = -eps / h**2
    return A

def solve_fd(eps=0.01, b=1.0, N=200):
    h = 1.0 / N
    x = np.linspace(0.0, 1.0, N+1)
    M = N - 1

    u_exact = exact_solution(x, eps, b)

    #Zentrale Differenzen
    A_c = build_central_matrix(eps, b, h, M)
    rhs_c = np.zeros(M)
    coeff_uN_c = -eps / h**2 + b / (2*h)
    rhs_c[-1] = -coeff_uN_c * 1.0
    u_interior_c = np.linalg.solve(A_c, rhs_c)
    u_c = np.zeros(N+1)
    u_c[0] = 0.0
```

```
    u_c[1:N] = u_interior_c
    u_c[N] = 1.0

    #Upwind
    A_u = build_upwind_matrix(eps, b, h, M)
    rhs_u = np.zeros(M)
    coeff_uN_u = -eps / h**2
    rhs_u[-1] = -coeff_uN_u * 1.0
    u_interior_u = np.linalg.solve(A_u, rhs_u)
    u_u = np.zeros(N+1)
    u_u[0] = 0.0
    u_u[1:N] = u_interior_u
    u_u[N] = 1.0

    return x, u_exact, u_c, u_u, A_c, A_u

def error_norms(u, u_ex):
    e = np.abs(u - u_ex)
    return np.max(e), np.sqrt(np.mean(e**2))

if __name__ == "__main__":
    eps = 1e-2
    b   = 1.0
    N   = 400

    x, u_ex, u_c, u_u, A_c, A_u = solve_fd(eps=eps, b=b, N=N)
    err_c = error_norms(u_c, u_ex)
    err_u = error_norms(u_u, u_ex)

    cond_c = np.linalg.cond(A_c)
    cond_u = np.linalg.cond(A_u)

    print(f"Zentrale FD:   $\\infty$-Fehler = {err_c[0]:.3e}, L2-Fehler =
    ↪  {err_c[1]:.3e}, Kond(A)={cond_c:.3e}")
    print(f"Upwind-Verf.:  $\\infty$-Fehler = {err_u[0]:.3e}, L2-Fehler =
    ↪  {err_u[1]:.3e}, Kond(A)={cond_u:.3e}")

    #Plot
    fig, ax = plt.subplots(figsize=(10,5))
    ax.plot(x, u_ex, '-', lw=2.0, label='Exakte Lösung')
    ax.plot(x, u_c, '--', lw=1.2, label='Zentrale FD')
    ax.plot(x, u_u, '-.', lw=1.2, label='Upwind')
    ax.set_xlabel('x')
    ax.set_ylabel('u(x)')
    ax.set_title(f'Vergleich ($\\epsilon$={eps}, b={b}, N={N})')
    ax.grid(alpha=0.4, ls='--')
    ax.legend(loc='upper left')

    #Zoom:
    left, right = 0.92, 1.0
    ix = np.where((x >= left) & (x <= right))[0]
```

```
    from mpl_toolkits.axes_grid1.inset_locator import inset_axes
    axins = inset_axes(ax, width="40%", height="60%", loc='lower right',
    ↪  borderpad=1.0)
    axins.plot(x[ix], u_ex[ix], '-', lw=2.0)
    axins.plot(x[ix], u_c[ix], '--', lw=1.2)
    axins.plot(x[ix], u_u[ix], '-.', lw=1.2)
    axins.set_xlim(left, right)
    axins.set_title('Zoom: Grenzschicht', fontsize=9)
    axins.grid(alpha=0.4, ls='--')

    plt.tight_layout()
    plt.show()
```

Python-Code zur Erzeugung von Abbildung 5.1

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({
    "font.size": 14,
    "axes.labelsize": 16,
    "axes.titlesize": 18,
    "legend.fontsize": 14,
    "xtick.labelsize": 14,
    "ytick.labelsize": 14
})


def exact_solution(x, eps):
    #stabile Form von u(x) = (exp(x/eps)-1)/(exp(1/eps)-1)
    alpha = np.exp(-1.0 / eps)
    return (np.exp((x - 1.0) / eps) - alpha) / (1.0 - alpha)

def solve_upwind(N, eps):
    h = 1.0 / N
    n = N - 1

    A = np.zeros((n, n), dtype=float)
    b = np.zeros(n, dtype=float)

    low = (-eps / h**2) - (1.0 / h)
    mid = (2.0 * eps / h**2) + (1.0 / h)
    up  = (-eps / h**2)

    #Matrix füllen
    for i in range(n):
        A[i, i] = mid
        if i - 1 >= 0:
            A[i, i-1] = low
```

```
        if i + 1 < n:
            A[i, i+1] = up

    u0, uN = 0.0, 1.0
    b[0]  -= low * u0
    b[-1] -= up  * uN

    u_inner = np.linalg.solve(A, b)

    u = np.zeros(N + 1)
    u[0] = u0
    u[1:N] = u_inner
    u[N] = uN
    return u

def solve_central(N, eps):
    h = 1.0 / N
    n = N - 1

    A = np.zeros((n, n), dtype=float)
    b = np.zeros(n, dtype=float)

    low = (-eps / h**2) - (1.0 / (2.0 * h))
    mid = (2.0 * eps / h**2)
    up  = (-eps / h**2) + (1.0 / (2.0 * h))

    for i in range(n):
        A[i, i] = mid
        if i - 1 >= 0:
            A[i, i-1] = low
        if i + 1 < n:
            A[i, i+1] = up

    u0, uN = 0.0, 1.0
    b[0]  -= low * u0
    b[-1] -= up  * uN

    u_inner = np.linalg.solve(A, b)

    u = np.zeros(N + 1)
    u[0] = u0
    u[1:N] = u_inner
    u[N] = uN
    return u

def einfty_error(u_num, x, eps):
    u_ex = exact_solution(x, eps)
    return np.max(np.abs(u_ex - u_num))

def main():
    eps = 1e-3
```

```
    Ns = [20, 40, 80, 160, 320, 640]

    hs, err_up, err_ce = [], [], []
    rows = []

    for N in Ns:
        h = 1.0 / N
        x = np.linspace(0.0, 1.0, N + 1)

        u_up = solve_upwind(N, eps)
        u_ce = solve_central(N, eps)

        e_up = einfty_error(u_up, x, eps)
        e_ce = einfty_error(u_ce, x, eps)

        hs.append(h)
        err_up.append(e_up)
        err_ce.append(e_ce)
        rows.append((N, h, e_up, e_ce))

    print("\nFehlervergleich (epsilon = 1e-3)")
    header = f"{'N':>6}  {'h=1/N':>12}  {'E_inf Upwind':>16}  {'E_inf
    ↪  Zentral':>16}"
    print(header)
    print("-" * len(header))
    for N, h, e_up, e_ce in rows:
        print(f"{N:6d}  {h:12.5e}  {e_up:16.5e}  {e_ce:16.5e}")

    #Plot
    hs = np.array(hs, dtype=float)
    err_up = np.array(err_up, dtype=float)
    err_ce = np.array(err_ce, dtype=float)

    plt.figure()
    plt.loglog(hs, err_up, marker="o", linewidth=1.5, label="Upwind:
    ↪  $E_\\infty$")
    plt.loglog(hs, err_ce, marker="s", linewidth=1.5, label="Zentral:
    ↪  $E_\\infty$")

    plt.gca().invert_xaxis()
    plt.xticks(hs, [f"{h:.4f}" for h in hs])
    plt.xlabel("$h=1/N$")
    plt.ylabel("$E_\\infty = \\max_i |u(x_i)-u_i|$")
    plt.title("Upwind vs. Zentraldifferenzen für $-\\varepsilon u'' + u' = 0$,
    ↪  $\\varepsilon=10^{-3}$")
    plt.grid(True, which="both")
    plt.legend()
    plt.tight_layout()
    plt.savefig("Einf_plot_epsilon1e-3.png", dpi=300)
    plt.show()
```

```python
if __name__ == "__main__":
    main()
```

Python-Code zur Erzeugung von Abbildung 5.3

```python
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({
    "font.size": 16,
    "axes.labelsize": 18,
    "axes.titlesize": 20,
    "legend.fontsize": 16,
    "xtick.labelsize": 15,
    "ytick.labelsize": 15
})

def rho(q):
    q = np.asarray(q, dtype=float)
    out = np.empty_like(q)

    small = np.abs(q) < 1e-8
    out[small] = 1.0

    qs = q[~small]
    out[~small] = qs * (np.cosh(qs) / np.sinh(qs))
    return out

def main():
    q = np.linspace(0.0, 6.0, 800)

    plt.figure(figsize=(4.8, 6.0))

    plt.plot(q, rho(q), linewidth=2.5,
             label=r"$\rho(q)=q\,\coth(q)$")

    plt.plot(q, q, ":", linewidth=2.0,
             label=r"Asymptotik: $\rho(q)\sim q$")

    plt.plot(q, 1 + q**2 / 3, "--", linewidth=2.0,
             label=r"$q\ll1$: $1+\frac{q^2}{3}$")

    plt.plot(q, 1 + q, "-.", linewidth=2.0,
             label=r"Vergleich: $\rho(q)=1+q$")

    plt.xlabel(r"$q$")
    plt.ylabel(r"$\rho(q)$")
    plt.title(r"Asymptotisches Verhalten der künstlichen Diffusion $\rho(q)$")
```

```
    plt.grid(True)
    plt.legend()
    plt.tight_layout()

    plt.savefig("rho_asymptotik.png", dpi=300, bbox_inches="tight")
    plt.show()

if __name__ == "__main__":
    main()
```

Python-Code zur Erzeugung von Abbildung 5.4

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker

plt.rcParams.update({
    "font.size": 14,
    "axes.labelsize": 16,
    "axes.titlesize": 18,
    "legend.fontsize": 14,
    "xtick.labelsize": 14,
    "ytick.labelsize": 14
})

def exact_solution(x, eps):
    alpha = np.exp(-1.0 / eps)
    return x - (np.exp(-(1.0 - x) / eps) - alpha) / (1.0 - alpha)

def solve_upwind(N, eps):
    h = 1.0 / N
    n = N - 1

    A = np.zeros((n, n))
    b = np.ones(n)

    low = -eps / h**2 - 1.0 / h
    mid =  2.0 * eps / h**2 + 1.0 / h
    up  = -eps / h**2

    for i in range(n):
        A[i, i] = mid
        if i > 0:
            A[i, i-1] = low
        if i < n-1:
            A[i, i+1] = up

```

```
    u = np.zeros(N + 1)
    u[1:N] = np.linalg.solve(A, b)
    return u

def solve_central(N, eps):
    h = 1.0 / N
    n = N - 1

    A = np.zeros((n, n))
    b = np.ones(n)

    low = -eps / h**2 - 1.0 / (2*h)
    mid =  2.0 * eps / h**2
    up  = -eps / h**2 + 1.0 / (2*h)

    for i in range(n):
        A[i, i] = mid
        if i > 0:
            A[i, i-1] = low
        if i < n-1:
            A[i, i+1] = up

    u = np.zeros(N + 1)
    u[1:N] = np.linalg.solve(A, b)
    return u

def solve_ilin_allen_southwell(N, eps, mu=1.0):
    h = 1.0 / N
    n = N - 1

    def coth(z):
        if abs(z) < 1e-12:
            return 1.0 / z + z / 3.0
        return 1.0 / np.tanh(z)

    theta = mu * h / (2.0 * eps)
    alpha = (1.0 / (2.0 * h)) * coth(theta)

    low = -(alpha + 1.0 / (2.0 * h))
    mid =  2.0 * alpha
    up  = -(alpha - 1.0 / (2.0 * h))

    A = np.zeros((n, n))
    b = np.ones(n)

    for i in range(n):
        A[i, i] = mid
        if i > 0:
            A[i, i-1] = low
        if i < n-1:
            A[i, i+1] = up
```

```
87
88      u = np.zeros(N + 1)
89      u[1:N] = np.linalg.solve(A, b)
90      return u
91
92  def einfty_error(u_num, x, eps):
93      return np.max(np.abs(u_num - exact_solution(x, eps)))
94
95  def main():
96      eps = 1e-3
97      Ns = [2, 4, 8, 16, 20, 40, 80, 160, 320, 640]
98
99      hs, err_up, err_ce, err_ias = [], [], [], []
100     rows = []
101
102     for N in Ns:
103         h = 1.0 / N
104         x = np.linspace(0.0, 1.0, N + 1)
105
106         e_up  = einfty_error(solve_upwind(N, eps), x, eps)
107         e_ce  = einfty_error(solve_central(N, eps), x, eps)
108         e_ias = einfty_error(solve_ilin_allen_southwell(N, eps), x, eps)
109
110         hs.append(h)
111         err_up.append(e_up)
112         err_ce.append(e_ce)
113         err_ias.append(e_ias)
114         rows.append((N, h, e_up, e_ce, e_ias))
115
116     print("\nFehlervergleich für -eps u'' + u' = 1, u(0)=u(1)=0 (epsilon =
        ↪  0.001)")
117     header = f"{'N':>6} {'h=1/N':>12} {'Upwind':>14} {'Zentral':>14}
        ↪  {'IAS':>14}"
118     print(header)
119     print("-" * len(header))
120     for N, h, e_up, e_ce, e_ias in rows:
121         print(f"{N:6d} {h:12.5e} {e_up:14.5e} {e_ce:14.5e} {e_ias:14.5e}")
122
123     hs = np.array(hs)
124     err_up = np.array(err_up)
125     err_ce = np.array(err_ce)
126     err_ias = np.array(err_ias)
127
128     plt.figure(figsize=(7.0, 5.0))
129
130     plt.loglog(hs, err_up, "-o", color="black", linewidth=1.5, markersize=6,
        ↪  label="Upwind")
131     plt.loglog(hs, err_ce, "--s", color="black", linewidth=1.5, markersize=6,
        ↪  label="Zentral")
132     plt.loglog(hs, err_ias, ":^", color="black", linewidth=1.5, markersize=6,
133                label="Il'in--Allen--Southwell")
```

```
    ax = plt.gca()
    ax.invert_xaxis()

    ax.xaxis.set_major_locator(mticker.LogLocator(base=10))
    ax.yaxis.set_major_locator(mticker.LogLocator(base=10))
    ax.xaxis.set_major_formatter(mticker.LogFormatterMathtext())
    ax.yaxis.set_major_formatter(mticker.LogFormatterMathtext())

    ax.grid(True, which="major", linestyle="--", linewidth=0.6, alpha=0.6)

    plt.xlabel(r"$h=1/N$")
    plt.ylabel(r"$E_\infty$")
    plt.legend(frameon=False)
    plt.tight_layout()

    plt.savefig("Einf_plot_eps1e-3_rhs1.pdf")
    plt.savefig("Einf_plot_eps1e-3_rhs1.png", dpi=300)
    plt.show()

if __name__ == "__main__":
    main()
```

Python-Code zur Erzeugung von Tabelle 5.1 und Abbildungen 5.5 und 5.5

```
import numpy as np
import matplotlib.pyplot as plt

def u_exact(x, eps):
    return (np.exp(x / eps) - 1.0) / (np.exp(1.0 / eps) - 1.0)

def grid_uniform(N):
    return np.linspace(0.0, 1.0, N + 1)

def grid_shishkin(N, eps):
    if N % 2 != 0:
        raise ValueError("N muss gerade sein (Shishkin-Gitter).")
    tau = min(0.5, 2.0 * eps * np.log(N))
    H = 2.0 * (1.0 - tau) / N
    h = 2.0 * tau / N

    x = np.zeros(N + 1)
    for i in range(N // 2 + 1):
        x[i] = i * H
    for i in range(N // 2 + 1, N + 1):
        x[i] = 1.0 - (N - i) * h
    return x, tau
```

```
23
24  def solve_upwind_nonuniform(x, eps):
25      N = len(x) - 1
26      h = np.diff(x)
27
28      u0, uN = 0.0, 1.0
29      A = np.zeros((N - 1, N - 1))
30      b = np.zeros(N - 1)
31
32      for i in range(1, N)
33          him = h[i - 1]
34          hip = h[i]
35          fac = 2.0 * eps / (him + hip)
36
37          cim1 = -(fac / him + 1.0 / him)
38          ci   =  (fac / hip + fac / him + 1.0 / him)
39          cip1 = -(fac / hip)
40
41          row = i - 1
42
43          if i == 1:
44              b[row] -= cim1 * u0
45          else:
46              A[row, row - 1] += cim1
47
48          A[row, row] += ci
49
50          if i == N - 1:
51              b[row] -= cip1 * uN
52          else:
53              A[row, row + 1] += cip1
54
55      u = np.zeros(N + 1)
56      u[0], u[N] = u0, uN
57      u[1:N] = np.linalg.solve(A, b)
58      return u
59
60  def einf_error(x, uh, eps):
61      return np.max(np.abs(u_exact(x, eps) - uh))
62
63  def make_error_table(eps, Ns):
64      rows = []
65      for N in Ns:
66          xU = grid_uniform(N)
67          uhU = solve_upwind_nonuniform(xU, eps)
68          EU = einf_error(xU, uhU, eps)
69
70          xS, tau = grid_shishkin(N, eps)
71          uhS = solve_upwind_nonuniform(xS, eps)
72          ES = einf_error(xS, uhS, eps)
73
```

```
        ratio = EU / ES if ES != 0 else np.inf
        rows.append((N, tau, EU, ES, ratio))
    return rows


def print_error_table(rows):
    wN, wT, wEU, wES, wR = 8, 14, 18, 18, 14

    print(f"{'N':>{wN}} {'tau':>{wT}} {'E_inf uniform':>{wEU}} {'E_inf
    ↪  Shishkin':>{wES}} {'ratio(U/S)':>{wR}}")
    print("-" * (wN + 1 + wT + 1 + wEU + 1 + wES + 1 + wR))

    for N, tau, EU, ES, ratio in rows:
        print(f"{N:{wN}d} {tau:{wT}.6e} {EU:{wEU}.6e} {ES:{wES}.6e}
        ↪  {ratio:{wR}.6e}")


def save_table_csv(rows, filename="error_table.csv"):
    with open(filename, "w", encoding="utf-8") as f:
        f.write("N,tau,E_inf_uniform,E_inf_shishkin,ratio_U_over_S\n")
        for N, tau, EU, ES, ratio in rows:
            f.write(f"{N},{tau:.16e},{EU:.16e},{ES:.16e},{ratio:.16e}\n")
    print(f"saved: {filename}")

def set_thesis_style():
    plt.rcParams.update({
        "figure.figsize": (8.5, 5.5),
        "figure.dpi": 120,
        "savefig.dpi": 400,
        "font.size": 18,
        "axes.titlesize": 22,
        "axes.labelsize": 20,
        "legend.fontsize": 16,
        "xtick.labelsize": 16,
        "ytick.labelsize": 16,
        "lines.linewidth": 2.6,
        "axes.grid": True,
        "grid.alpha": 0.25,
    })


def save_fig(name):
    plt.tight_layout()
    plt.savefig(name + ".pdf", bbox_inches="tight")
    plt.savefig(name + ".png", bbox_inches="tight")
    print(f"saved: {name}.pdf / {name}.png")

def zoom_window(eps, N, tau):
    width = max(6.0 * eps, 4.0 * eps * np.log(N))
    xl = min(1.0 - width, 1.0 - 1.2 * tau)
    return max(0.0, xl), 1.0
```

```python


def plot_solution_zoom(N, eps):
    xU = grid_uniform(N)
    uU = solve_upwind_nonuniform(xU, eps)
    uE = u_exact(xU, eps)

    xS, tau = grid_shishkin(N, eps)
    uS = solve_upwind_nonuniform(xS, eps)

    xl, xr = zoom_window(eps, N, tau)

    plt.figure()
    plt.plot(xU, uE, label="exakt")
    plt.plot(xU, uU, "--", label="Upwind (uniform)")
    plt.plot(xS, uS, "-.", label="Upwind (Shishkin)")
    plt.xlim(xl, xr)
    plt.xlabel(r"$x$")
    plt.ylabel(r"$u(x)$")
    plt.title(rf"Grenzschicht-Zoom bei $x=1$ ($\varepsilon={eps:g}$, $N={N}$)")
    plt.legend(loc="best")
    save_fig(f"solution_zoom_N{N}")

def plot_einf_convergence(rows, eps):
    Ns = np.array([r[0] for r in rows], dtype=float)
    EU = np.array([r[2] for r in rows], dtype=float)
    ES = np.array([r[3] for r in rows], dtype=float)

    plt.figure()
    plt.loglog(Ns, EU, "o--", label=r"$E_\infty$ (uniform)")
    plt.loglog(Ns, ES, "o-",  label=r"$E_\infty$ (Shishkin)")
    plt.xlabel(r"$N$")
    plt.ylabel(r"$E_\infty$")
    plt.title(rf"Maximaler Fehler $E_\infty(N)$ ($\varepsilon={eps:g}$)")
    plt.legend(loc="best")
    save_fig(f"einf_convergence_eps{eps:g}")

def main():
    set_thesis_style()

    eps = 1e-2
    Ns = [8, 16, 32, 64, 128, 256, 512, 1024]  # gerade
    N_show = 256

    # (2) Fehlertabelle
    rows = make_error_table(eps, Ns)
    print_error_table(rows)
    save_table_csv(rows, "error_table.csv")

    # (1) Grenzschicht-Zoom-Plot der Lösung
    plot_solution_zoom(N_show, eps)
```

```
174
175     # (3) Plot des maximalen Fehlers E_inf(N)
176     plot_einf_convergence(rows, eps)
177
178     plt.show()
179
180
181 if __name__ == "__main__":
182     main()
```