Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Projekt Zeitaufgelöste Photolumineszenz: Angewandte Mathematik (M1)

Name: Aaron Zielstorff
Mtr.Nr.: 567183

Fachbereich: FB1
Studiengang: M. Elektrotechnik
Fachsemester: 1. FS
Fach: M1 Angewandte Mathematik
Dozent: Prof. Dr. A. Zeiser
Abgabe am: 20. März 2022

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Perowskit-Solarzellen sind der neue Stern am Himmel der Photovoltaik. Innerhalb von wenigen Jahren sind diese neuartigen Materialien zu konkurrenzfähigen Dünnschicht-Solarzellen mit weit über 20% Wirkungsgrad erwachsen. Daraus begründen sich die enorme Popularität und auch die vielen Hoffnungen an diese Halbleiter-Materialklasse. Dennoch gibt es noch zahlreiche Probleme und ungeklärte Fragestellungen, die die Forschung adressieren kann. Andreas Bartelt und seine Gruppe untersuchen an der HTW Berlin diese Materialien mithilfe von zeitaufgelösten Photolumineszenz-Spektren (TRPL, *time resolved photo- luminescence*): das Material wird mithilfe eines Lasers angeregt und die aus der Anregung entstehende Aussendung von Licht in Abhängigkeit von der Zeit gemessen.
Um aus den experimentellen Daten Rückschlüsse auf Lebensdauern und Hinweise auf effizienzlimitierende Prozesse in den neuen Materialien zu ziehen, ist eine Simulation der Dynamiken im Halbleiter notwendig. In dieser Hausarbeit sollen Sie die Grundlagen einer solchen Simulation entwickeln.
Der Halbleiter der Dicke d zwischen zwei Materialien wird gleichmäßig in der Fläche bestrahlt und erzeugt so eine Ladungsträgerdichte u im Halbleiter (siehe Abbildung 1). Die Abhängigkeit von u in der Ebene senkrecht zur Dicke kann in guter Näherung vernachlässigt werden, so dass die Ladungsträgerdichte eine Funktion der Zeit t und der Tiefe z ist. Die Elektronendynamik im Halbleiter nach der Bestrahlung wird durch eine Diffusionsgleichung modelliert [3]. Die Ladungsträgerdichte $u(t, z)$ im Halbleiter genügt der parabolischen Differenzialgleichung

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial z^2} - (k_1 + k_2 N_D)u - k_2 u^2 + s(t, z), \quad t \geq t_0, 0 < z < d \tag{1}$$

mit der Anfangsbedingung

$$u(t_0, z) = 0, \quad 0 \geq z \geq d \tag{2}$$

und den Randbedingungen

$$D\frac{\partial u}{\partial z}(t, 0) = S_L u(t, 0), \quad D\frac{\partial u}{\partial z}(t, d) = -S_R u(t, d), \quad t \geq t_0 \tag{3}$$

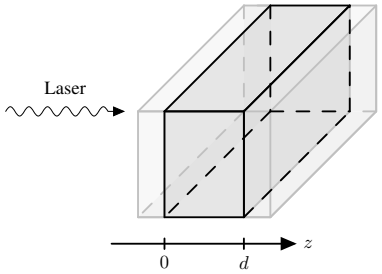

Abb. 1: Skizze des Aufbaus der Probe. Der Halbleiter aus Peroskit liegt zwischen zwei Materialien und wird in der Ebene senkrecht zu z als unendlich ausgedehnt angenommen. Die Bestrahlung mit Laserlicht erfolgt gleichmäßig aus der Richtung $z < 0$.

Hier ist $s(t, z)$ die Ladungsträgerdichte, die durch die Bestrahlung pro Zeiteinheit erzeugt wird, D die Diffusionskonstante, N_D die Dotierungsdichte, k_1 bzw. k_2 die Rekombinationskonstanten (Shockley Read Hall Rekombination, bzw. direkte Rekombination) und α die Absorptionskonstante. Die Konstanten S_L $(z = 0)$ bzw. S_R $(z = d)$ bestimmen die Rekombinationsraten an den jeweiligen Grenzschichten.

In dieser Arbeit sollen folgende Parameter verwendet werden

Parameter	d	D	K_1	k_2	N_D	S_L	S_R	α
Einheit	$[\mu m]$	$\left[\frac{cm^2}{s}\right]$	$\left[\frac{1}{s}\right]$	$\left[\frac{cm^3}{s}\right]$	$\left[\frac{1}{cm^3}\right]$	$\left[\frac{cm}{s}\right]$	$\left[\frac{cm}{s}\right]$	$\left[\frac{1}{cm}\right]$
Wert	0.3	0.003	10^6	10^{-8}	10^{15}	10	10^5	10^5

Tab. 1: Verwendete Parameter

Achtung: Verwenden Sie in der Simulation durchgehend μm und μs als Einheit!

Die Simulation von Gleichung 1 wird schrittweise entwickelt. In Abschnitt 2 wird zunächst die Zeitabhängigkeit vernachlässigt und eine Diskretisierung im Raum der stationären Gleichung mittels finiter Differenzen entwickelt. Der folgende Abschnitt 3 untersucht eine Klasse von numerischen Verfahren zur Lösung von Anfangswertproblemen, die besonders gut für den vorliegenden Fall geeignet sind. Im letzten Abschnitt 4 werden beide Ansätze mithilfe der Linienmethode kombiniert und eine numerische Lösung der Ausgangsgleichung berechnet.

2 Finite Differenzen der stationären Gleichung

In diesem Teil der Arbeit soll die stationäre Verteilung der Ladungsträger, d. h. für den Fall $\partial u/\partial t = 0$, bei kontinuierlicher Bestrahlung modelliert werden. In diesem Fall ist die Gleichung durch

$$D\frac{\partial^2 u}{\partial z^2} - (k_1 + k_2 N_D)u - k_2 u^2 = -s(z), \quad 0 < z < d \tag{4}$$

mit Randbedingungen

$$D\frac{\partial u}{\partial z}(0) = S_L u(0), \quad D\frac{\partial u}{\partial z}(d) = -S_R u(d) \tag{5}$$

gegeben. Hier ist $s(z)$ die Ladungsträgerdichte, die pro Zeiteinheit durch die externe Quelle erzeugt wird.

Die Gleichung soll näherungsweise mithilfe der Finite Differenzen Methode gelöst werden [4] (benötigte Abschnitte werden auf Moodle bereitgestellt). Dazu wird das Gebiet $z \in [0, d]$ in M gleich große Intervalle der Länge h aufgeteilt und die Knotenpunkte mit $0 = z_0 < z_1 < ... < z_N = d$ bezeichnet. Die genäherte Ladungsdichte an den Stellen z_i wird mit u_i bezeichnet, d. h. es soll

$$u_i \approx u(z_i), \quad i = 0, 1, ..., N$$

gelten. Für diese Werte wird ein Gleichungssystem hergeleitet und anschließend numerisch gelöst. Zwischenzeitlich werden Hilfsknoten an den Stellen $z_{-1} = -h$ und $z_{N+1} = d + h$ mit Werten u_{-1} und u_{N+1} verwendet, die jedoch später durch die Randbedingungen eliminiert werden.

2.1 Lineare stationäre Gleichung

Im ersten Schritt soll nur der in u lineare Anteil der Gleichung 4 ohne den quadratischen Term $-k_2 u^2$ untersucht werden:

$$D\frac{\partial^2 u}{\partial z^2} - ku = -s(z), \quad 0 < z < d \tag{6}$$

mit $k = k_1 + k_2 N_D$. Die Randbedingungen sind weiterhin durch Gleichung 5 gegeben.

Die Behandlung der stationären Gleichung erfolgt ähnlich zu der in [4, Abschnitt 8.8] beschriebenen Methode. Jedoch werden die Randbedingungen unterschiedlich behandelt. Folgende Aufgaben wurden bearbeitet.

2.1.1 Methode aus [4, Abschnitt 8.8] für Anwendung auf Gleichung 6

Die beschriebene Methode der finiten Differenzen für Zweipunkt-Randwertprobleme ermöglicht die Lösung von linearen Differenzialgleichungen zweiter Ordnung. Die Grundlage der finiten Differenzen Methode ist das Aufstellen von diskreten Gleichungen durch das Substituieren der Ableitungen mit geeigneten finiten Differenzen. Das Ergebnis ist eine Matrix, mit der in unserem konkreten Beispiel die zeitabhängige Stelle der Ladungsträgerdichte u_i mathematisch beschrieben werden kann.

Für die beschriebene Methode sind drei Schritte notwendig:

1. Diskretisierung des Bereiches $z \in [0, d]$ in M gleich große Intervalle der Länge h mit $h = \frac{d-0}{N}$.
2. Diskretisierung der Differenzialgleichung an den Knotenpunkten $u_1, ..., u_{N-1}$ mit den Approximationen der Ableitungen.
3. Diskretisierung der Randbedingungen und Aufstellen des Gleichungssystems durch die Elimination von u_{-1} und u_{N+1}.

2.1.2 Herleitung der Gleichung für die gesuchten Werte $u_0, ..., u_N$

Approximation der 1. Ableitung:

$$\frac{\partial u}{\partial z} = \frac{u_{i+1} - u_{i-1}}{2h}$$

Approximation der 2. Ableitung:

$$\frac{\partial^2 u}{\partial z^2} = \frac{u_{i+1} - 2 \cdot u_i + u_{i-1}}{h^2}$$

Durch das Einsetzen der Approximationen der beiden Ableitungen ergeben sich für u_i mit $i = 0, ..., N$ unter der der Festlegung $s_i = s(z_i)$ und $u_i \approx u(z_i)$ folgende Gleichungen:

$$
\begin{aligned}
D \cdot \frac{u_1 - 2 \cdot u_0 + u_{-1}}{h^2} - k \cdot u_0 &= -s_0 \quad mit \quad i = 0 \\
D \cdot \frac{u_{i+1} - 2 \cdot u_i + u_{i-1}}{h^2} - k \cdot u_i &= -s_i \\
D \cdot \frac{u_{N+1} - 2 \cdot u_N + u_{N-1}}{h^2} - k \cdot u_N &= -s_N \quad mit \quad i = N
\end{aligned}
$$

Dabei ist u_i die approximierte Ladungsträgerdichte an der Stelle z_i und s_i die Ladungsträgerdichte, die durch die Bestrahlung mit dem Laser an der Stelle z_i auftritt.

2.1.3 Approximation der Ableitungen an den Randbedingungen

Zunächst werden die ersten Ableitungen an den Randbedingungen (Gleichung 5) approximiert durch:

$$
u'(0) \approx \frac{u_1 - u_{-1}}{2h}, \quad u'(d) \approx \frac{u_{N+1} - u_{N-1}}{2h}
$$

Durch das Auflösen nach u_{-1} bzw. u_{N+1} ergibt sich:

$$
u_{-1} = -(u'(0) \cdot 2h) + u_1, \quad u_{N+1} = u'(d) \cdot 2h + u_{N-1}
$$

Einsetzen an den Knoten z_0 und z_N ergibt:

$$
\begin{aligned}
D * \frac{u_1 - 2u_0 - (u'_0 \cdot 2h) + u_1}{h^2} - k \cdot u_0 &= -s_0 \quad mit \quad i = 0 \\
D * \frac{u'_N \cdot 2h + u_{N-1} - 2u_N + u_{N-1}}{h^2} - k \cdot u_N &= -s_N \quad mit \quad i = N
\end{aligned}
$$

Nun können die Randbedingungen aus Gleichung 5 nach den 1. Ableitungen umgestellt und eingesetzt werden:

$$
u'(0) = \frac{S_L \cdot u_0}{D}, \quad u'(N) = \frac{-S_R \cdot u_N}{D}
$$

$$D * \frac{u_1 - 2u_0 - (\frac{S_L \cdot u_0}{D} \cdot 2h) + u_1}{h^2} - k \cdot u_0 = -s_0 \quad mit \quad i = 0$$
$$D * \frac{\frac{-S_R \cdot u_N}{D} \cdot 2h + u_{N-1} - 2u_N + u_{N-1}}{h^2} - k \cdot u_N = -s_N \quad mit \quad i = N$$

Die Gleichungen an den Stellen $i = 0, i, N$ werden nun vereinfacht bzw. umgeformt:

$$\frac{D}{h^2} \cdot \left[2u_1 - \left(2 + \frac{s_L \cdot 2h}{D} + \frac{h^2 \cdot k}{D}\right) \cdot u_0\right] = -s_0 \quad mit \quad i = 0$$
$$\frac{D}{h^2} \cdot \left[1u_{i+1} - \left(2 + \frac{h^2 \cdot k}{D}\right) \cdot u_i + 1u_{i-1}\right] = -s_i$$
$$\frac{D}{h^2} \cdot \left[\left(\frac{-s_R \cdot 2h}{D} - 2 - \frac{h^2 \cdot k}{D}\right) \cdot u_N + 2u_{N-1}\right] = -s_N \quad mit \quad i = N$$

2.1.4 Aufstellen des linearen Gleichungssystems

Aus den soeben Aufgestellten Gleichungen bzw. den grün eingefärbten Termen wird für die Größen $u_0, ..., u_N$ analog zu [4, 8.133] das lineare Gleichungssystem aufgestellt. Dabei soll das Gleichungssystem folgende Form aufweisen:

$$Au = b, \quad u = [u_i], \quad b = [b_i] \tag{7}$$

Die Gleichungen werden analog zu den Knotenpunkten $z_0, ..., z_N$ geordnet.

Für die Koeffizientenmatrix ergibt sich:

$$A = \frac{D}{h^2} \cdot \begin{bmatrix} \left(2 + \frac{s_L \cdot 2h}{D} + \frac{h^2 \cdot k}{D}\right) & 2 & & & & \\ 1 & \left(2 + \frac{h^2 \cdot k}{D}\right) & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & \left(2 + \frac{h^2 \cdot k}{D}\right) & 1 & \\ & & & 2 & \left(\frac{-s_R \cdot 2h}{D} - 2 - \frac{h^2 \cdot k}{D}\right) & \end{bmatrix} \cdot \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix}$$

$$A = - \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{N-1} \\ s_N \end{bmatrix}$$

2.1.5 Matlab Routine zum Berechnen der Matrix A

Die Routine kann unter dem Dokumentennamen `fd_lin_matrix.m` gefunden werden im beigefügten Ordner.

2.1.6 Matlab Routine zum Lösen des Gleichungssystems (Gleichung 7)

Die Routine kann unter dem Dokumentennamen `stationär_lin.m` gefunden werden im beigefügten Ordner.

2.1.7 Testen der Matlab Routine

Zum Testen der Routine wird eine Funktion $u(z)$ wie folgt vorgegeben:

$$u(z) = a \cdot e^{\lambda z}$$

Weiterhin werden die Konstanten d, D, k, a und λ wie folgt definiert:

$$
\begin{aligned}
d &= 0,3\mu m \\
D &= 0,003\frac{cm^2}{s} \\
k_1 &= 10^6\frac{1}{s} \\
k_2 &= 10^{-8}\frac{cm^3}{s} \\
N_D &= 10^{15}\frac{1}{cm^3} \\
k &= k_1 + k_2 \cdot N_D \\
a &= 1 \\
\lambda &= 1
\end{aligned}
$$

Nachfolgend muss die erste und zweite Ableitung der Funktion $u(z)$ ermittelt werden, um eine geeignete rechte Seite s zu berechnen:

$$
\begin{aligned}
u'(z) &= \lambda \cdot a \cdot e^{\lambda z} \\
u''(z) &= \lambda^2 \cdot a \cdot e^{\lambda z}
\end{aligned}
$$

Die rechte Seite s ergibt sich zu:

$$
s = -\left(D \cdot u''(z) - k \cdot u(z)\right)
$$

Abschließend berechnen sich die Konstanten S_L und S_R zu:

$$
\begin{aligned}
S_L &= D \cdot \lambda \\
S_R &= -D \cdot \lambda
\end{aligned}
$$

Das Matlab Skript mit den Konstanten kann im beigefügten Ordner unter dem Dokumententitel `konstanten.m` gefunden werden.

Abbildung 2 zeigt die erfolgreiche Testung der Routine Anhand des Vergleiches zwischen der analytischen Lösung und der der Lösung über die implementierte Routine. Anhand des unteren Abschnittes der Grafik kann festgehalten werden, dass die Lösung über die Routine

der realen Lösung annähernd entspricht. Der größe Fehler tritt an den Randstellen auf.

Weiterhin zeigt Abbildung 3, dass der absolute Fehler der Methode mit steigender Anzahl an Teilintervallen N abnimmt. Außerdem kann abgelesen werden, dass es sich um ein Verfahren zweiter Ordnung handelt, da der Fehler um zwei Dekaden sinkt, wenn die Anzahl der Teilintervalle um eine Dekade erhöht wird.

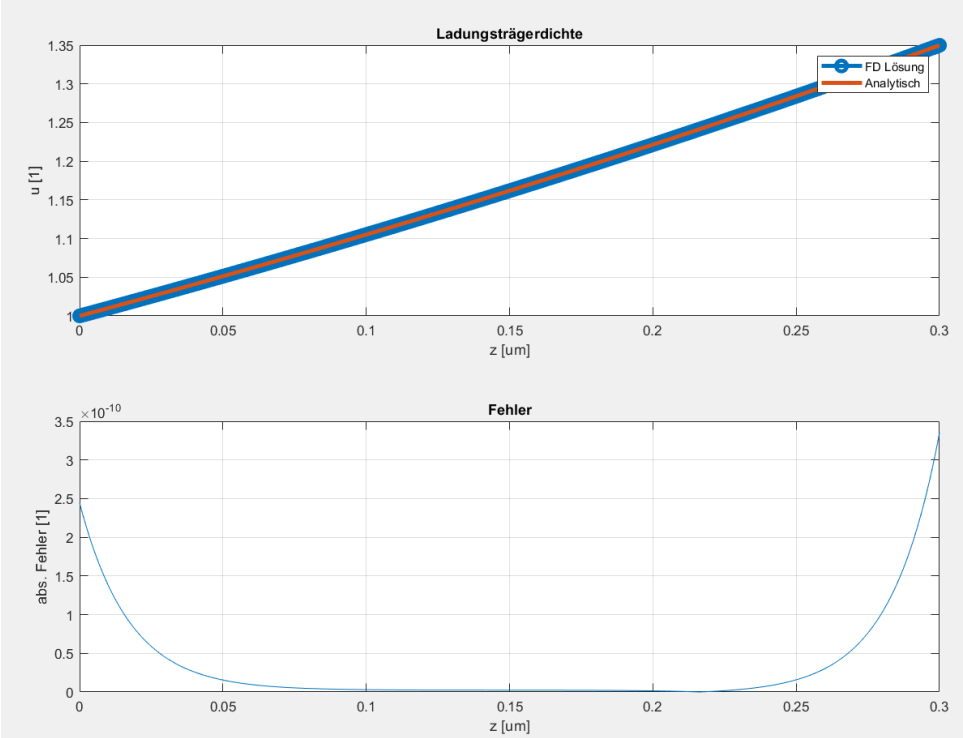

Abb. 2: Vergleich der Analytischen Lösung mit der Lösung über die entwickelte Routine, sowie Darstellung des Fehlers der Lösung mittels der Routine

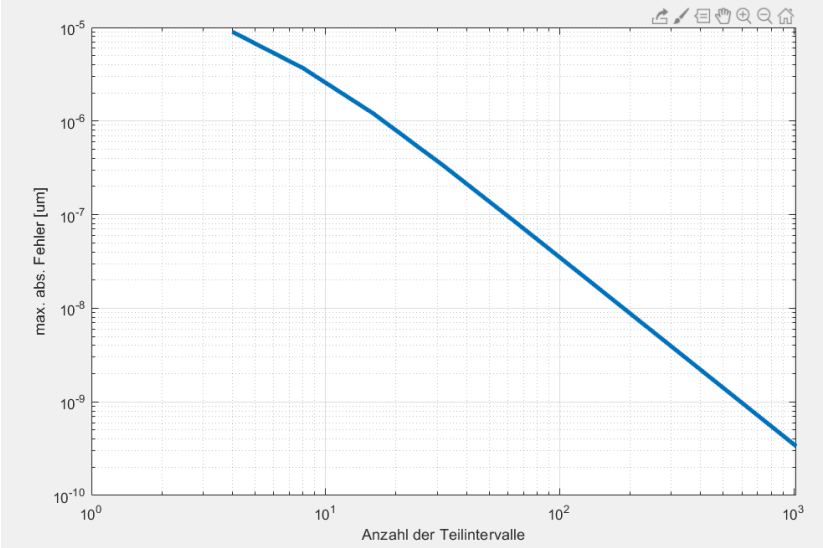

Abb. 3: Darstellung zur Ermittlung der Ordnung der Methode

Die genutzte Matlab Routine zum Testen der Routine kann unter dem Dokumentennamen `ordnung_stationär_lin.m` gefunden werden im beigefügten Ordner.

2.1.8 Anwendung der Routine auf spezielle Fälle

In dieser Teilaufgabe soll mit der entwickelten Routine die Lösung für die Fälle

$$s(z) = S_0 \cdot e^{-\alpha z}, \quad S_0 = 10^2, 10^3, 10^4 \frac{1}{\mu m^3 \mu s}$$

berechnet werden. Weiterhin soll ein geeignetes N so bestimmt werden, dass der relative Fehler in u_i maximal 1‰ beträgt.

Das Ergebnis für die drei Fälle von S_0 ist in Abbildung 4 zu erkennen.

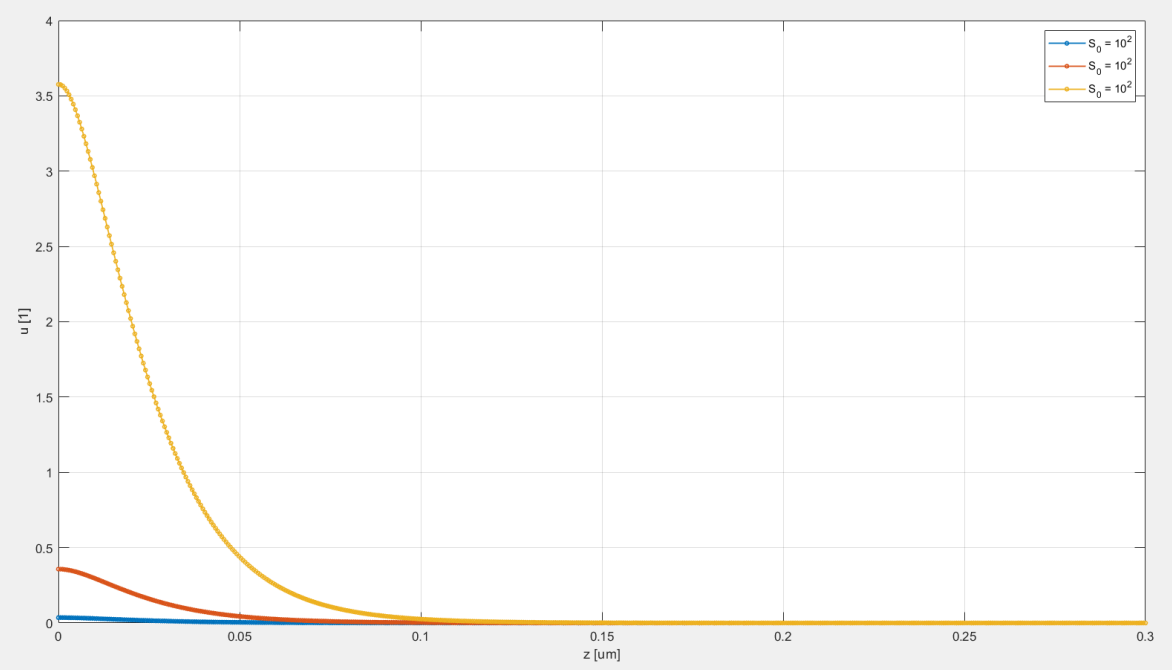

Abb. 4: Lösung von $s(z)$ für verschiedene S_0

Um ein geeignetes N zu bestimmen, um den realtiven Fehler zu erreichen, wird das N in einer Schleife iterativ erhöht. Dabei wird der Fehler relativ zum vorherigen N berechnet. Das ergebnis dieser Routine wird in Abbildung 5 dargestellt.
Die Routine kann unter dem Dokumentennamen `exper_stationär_lin.m` gefunden werden im beigefügten Ordner. Es kann festgehalten werden, dass das N als 512 gewählt werden muss, um einen relativen Fehler in u_i von maximal 1‰ zu erreichen.

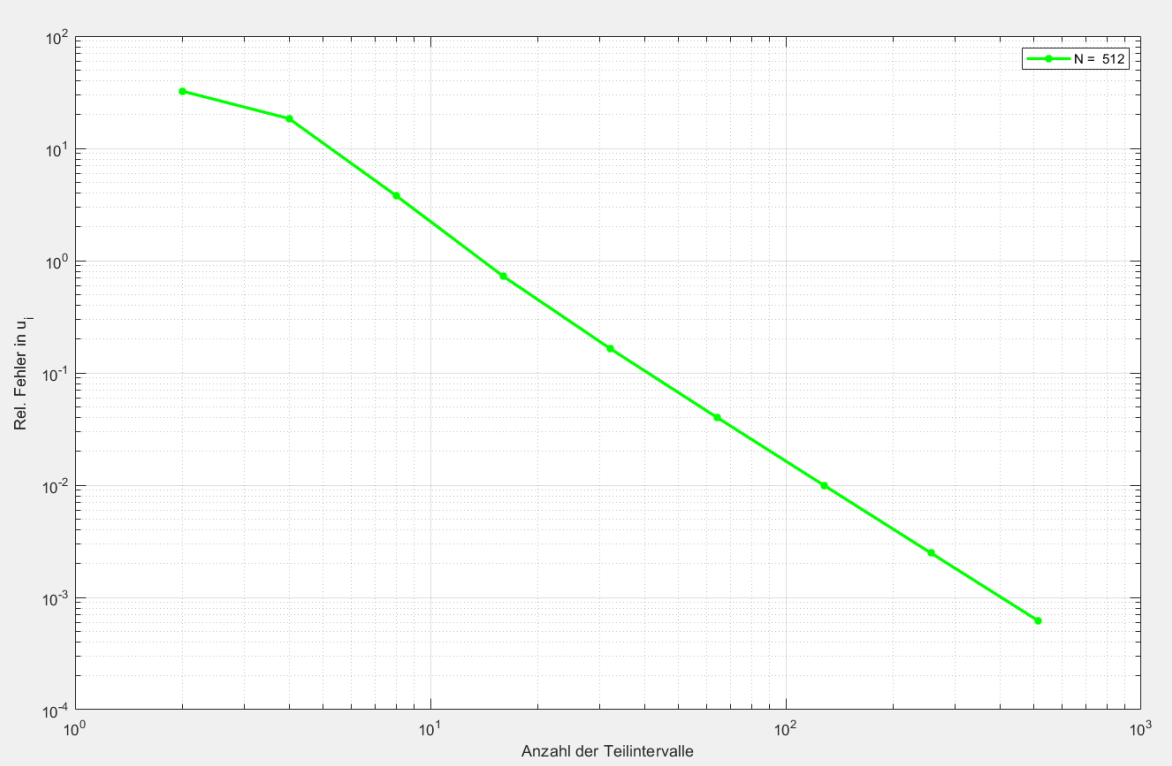

Abb. 5: Experimentelle Ermittlung von N, so dass der relative Fehler in u_i maximal 10^{-3} beträgt

2.2 Nichtlineare stationäre Gleichung

Im zweiten Schritt soll nun die volle nichtlineare Gleichung Gleichung 4 mit Randbedingung Gleichung 5 gelöst werden. Analog zum letzten Abschnitt wird ein nichtlineares Gleichungssystem aufgestellt, das mithilfe des Newton-Verfahrens aus der Belegarbeit gelöst werden soll.

2.2.1 Herleitung der nichtlinearen Gleichungen für die gesuchten Werte $u_0, ..., u_N$

Die nichtlinearen Gleichungen sollen in der Form

$$F(u) = b \tag{8}$$

dargestellt werden. Wie auch schon im linearen Fall werden die Randbedingungen berücksichtigt.

Zunächst werden wieder die Approximationen der 1. und 2. Ableitung aufgestellt.

Approximation der 1. Ableitung:

$$\frac{\partial u}{\partial z} = \frac{u_{i+1} - u_{i-1}}{2h}$$

Approximation der 2. Ableitung:

$$\frac{\partial^2 u}{\partial z^2} = \frac{u_{i+1} - 2 \cdot u_i + u_{i-1}}{h^2}$$

Durch das Einsetzen der Approximationen der beiden Ableitungen ergeben sich für u_i mit $i = 0, ..., N$ unter der der Festlegung $s_i = s(z_i)$ und $u_i \approx u(z_i)$ folgende Gleichungen:

$$\begin{aligned}
D \cdot \frac{u_1 - 2 \cdot u_0 + u_{-1}}{h^2} - k \cdot u_0 - k_2 \cdot u_0^2 &= -s_0 \quad mit \quad i = 0 \\
D \cdot \frac{u_{i+1} - 2 \cdot u_i + u_{i-1}}{h^2} - k \cdot u_i - k_2 \cdot u_0^2 &= -s_i \\
D \cdot \frac{u_{N+1} - 2 \cdot u_N + u_{N-1}}{h^2} - k \cdot u_N - k_2 \cdot u_0^2 &= -s_N \quad mit \quad i = N \\
mit \quad k = k_1 + k_2 \cdot N_D
\end{aligned}$$

Anschließend werden die ersten Ableitungen an den Randbedingungen (Gleichung 5) approximiert durch:

$$u'(0) \approx \frac{u_1 - u_{-1}}{2h}, \quad u'(d) \approx \frac{u_{N+1} - u_{N-1}}{2h}$$

Durch das Auflösen nach u_{-1} bzw. u_{N+1} ergibt sich:

$$u_{-1} = -(u'(0) \cdot 2h) + u_1, \quad u_{N+1} = u'(d) \cdot 2h + u_{N-1}$$

Einsetzen an den Knoten z_0 und z_N ergibt:

$$\begin{aligned}
D * \frac{u_1 - 2u_0 - (u'_0 \cdot 2h) + u_1}{h^2} - k \cdot u_0 - k_2 \cdot u_0^2 &= -s_0 \quad mit \quad i = 0 \\
D * \frac{u'_N \cdot 2h + u_{N-1} - 2u_N + u_{N-1}}{h^2} - k \cdot u_N - k_2 \cdot u_0^2 &= -s_N \quad mit \quad i = N
\end{aligned}$$

Nun können die Randbedingungen aus Gleichung 5 nach den 1. Ableitungen umgestellt und eingesetzt werden:

$$u'(0) = \frac{S_L \cdot u_0}{D}, \quad u'(N) = \frac{-S_R \cdot u_N}{D}$$

$$D * \frac{u_1 - 2u_0 - (\frac{S_L \cdot u_0}{D} \cdot 2h) + u_1}{h^2} - k \cdot u_0 - k_2 \cdot u_0^2 = -s_0 \quad mit \quad i = 0$$
$$D * \frac{\frac{-S_R \cdot u_N}{D} \cdot 2h + u_{N-1} - 2u_N + u_{N-1}}{h^2} - k \cdot u_N - k_2 \cdot u_0^2 = -s_N \quad mit \quad i = N$$

Die Gleichungen an den Stellen $i = 0, i, N$ werden nun vereinfacht bzw. umgeformt:

$$\frac{D}{h^2} \cdot \left[2u_1 - \left(\frac{2D + S_L \cdot 2h + h^2 k}{D} + \frac{h^2 \cdot k_2}{D} \cdot u_0\right) \cdot u_0\right] = -s_0 \quad mit \quad i = 0$$
$$\frac{D}{h^2} \cdot \left[1u_{i+1} - \left(\frac{2D + h^2 k}{D} + \frac{h^2 \cdot k_2}{D} \cdot u_i\right) \cdot u_i + 1u_{i-1}\right] = -s_i$$
$$\frac{D}{h^2} \cdot \left[\left(\frac{-S_R \cdot 2h - 2D - h^2 k}{D} - \frac{h^2 \cdot k_2}{D} \cdot u_N\right) \cdot u_N + 2u_{N-1}\right] = -s_N \quad mit \quad i = N$$

Die grün makierten Terme werden später für die Jacobi-Matrix benötigt.

Abschließend können die Gleichungen in der Form wie in Gleichung 14 gefordert dargestellt werden:

$$\begin{bmatrix} F_0 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix} = \frac{D}{h^2} \cdot \begin{bmatrix} -\left(\frac{2D+S_L \cdot 2h+h^2k}{D} + \frac{h^2 \cdot k_2}{D} \cdot u_0\right) \cdot u_0 + 2u_1 \\ \vdots \\ u_{i-1} - \left(\frac{2D+h^2k}{D} + \frac{h^2 \cdot k_2}{D} \cdot u_i\right) \cdot u_i + u_{i+1} \\ \vdots \\ 2u_{N-1} + \left(\frac{-S_R \cdot 2h-2D-h^2k}{D} - \frac{h^2 \cdot k_2}{D} \cdot u_N\right) \cdot u_N \end{bmatrix}$$

$$= - \begin{bmatrix} s_0 \\ \vdots \\ s_i \\ \vdots \\ s_N \end{bmatrix}$$

2.2.2 Matlab Routine zur Berechnung von F der letztem Teilaufgabe

Die Routine kann unter dem Dokumentennamen `fd_nonlin.m` gefunden werden im beigefügten Ordner.

2.2.3 Berechnung der Jacobi-Matrix DF von F bezüglich u

Allgemein kann die Jacobi-Matrix DF von F bezüglich u wie folgt dargestellt werden:

$$DF(u) = \begin{bmatrix} \frac{\partial F_0(u)}{\partial u_0} & \cdots & \frac{\partial F_0(u)}{\partial u_i} & \cdots & \frac{\partial F_0(u)}{\partial u_N} \\ \vdots & & \vdots & & \vdots \\ \frac{\partial F_i(u)}{\partial u_0} & \cdots & \frac{\partial F_i(u)}{\partial u_i} & \cdots & \frac{\partial F_i(u)}{\partial u_N} \\ \vdots & & \vdots & & \vdots \\ \frac{\partial F_N(u)}{\partial u_0} & \cdots & \frac{\partial F_N(u)}{\partial u_i} & \cdots & \frac{\partial F_N(u)}{\partial u_N} \end{bmatrix}$$

Mit Hilfe der grün makierten Terme aus Unterunterabschnitt 2.2.1 kann nun die konkrete Jacobi-Matrix DF aufgestellt werden.

$$DF(u) = \frac{D}{h^2} \cdot \begin{bmatrix} -\left(\frac{2D+S_L \cdot 2h+h^2k}{D} + \frac{h^2 \cdot k_2}{D} \cdot u_0\right) & 2 & & & \\ 1 & -\left(\frac{2D+h^2k}{D} + \frac{h^2 \cdot k_2}{D} \cdot u_i\right) & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -\left(\frac{2D+h^2k}{D} + \frac{h^2 \cdot k_2}{D} \cdot u_i\right) & 1 \\ & & & 2 & \left(\frac{-S_R \cdot 2h-2D-h^2k}{D} - \frac{h^2 \cdot k_2}{D} \cdot u_N\right) \end{bmatrix}$$

2.2.4 Matlab Routine zur Berechnung von DF der letztem Teilaufgabe

Die Routine kann unter dem Dokumentennamen `fd_nonlin_jac.m` gefunden werden im beigefügten Ordner.

2.2.5 Matlab Routine zur Lösung des Gleichungssystems (Gleichung 14)

Über die *Deal*-Funktion von Matlab kann ein Funktionshandle mit den Routinen für F und DF erstellt werden.
Als Startwert dient ein Spaltenvekor mit $N+1$ Zeilen die jeweils eine 0 enthalten.
Das Newton-Verfahren zum Lösen Gleichungssystems bekommt den Funktionshandle $f(u)$, den Startwert, die geforderte Toleranz und die maximale anzahl an Schritten übergeben.
Die Routine kann unter dem Dokumentennamen `stationaer_nonlin.m` gefunden werden im beigefügten Ordner.

2.2.6 Testen der Matlab Routine

Zum Testen der Routine wird eine Funktion $u(z)$ wie auch schon in Unterunterabschnitt 2.1.5 vorgegeben:

$$u(z) = a \cdot e^{\lambda z}$$

Weiterhin werden die Konstanten d, D, k, a und λ wie folgt definiert:

$$\begin{aligned} d &= 0,3\mu m \\ D &= 0,003\frac{cm^2}{s} \\ k_1 &= 10^6\frac{1}{s} \\ k_2 &= 10^{-8}\frac{cm^3}{s} \\ N_D &= 10^{15}\frac{1}{cm^3} \\ k &= k_1 + k_2 \cdot N_D \\ a &= 1 \\ \lambda &= 1 \end{aligned}$$

Nachfolgend muss die erste und zweite Ableitung der Funktion $u(z)$ ermittelt werden, um eine geeignete rechte Seite s zu berechnen:

$$u'(z) = \lambda \cdot a \cdot e^{\lambda z}$$
$$u''(z) = \lambda^2 \cdot a \cdot e^{\lambda z}$$

Die rechte Seite s ergibt sich zu:

$$s = -\left(D \cdot u''(z) - k \cdot u(z) - k_2 \cdot u(z)^2\right)$$

Abschließend berechnen sich die Konstanten S_L und S_R zu:

$$S_L = D \cdot \lambda$$
$$S_R = -D \cdot \lambda$$

Das Matlab Skript mit den Konstanten kann im beigefügten Ordner unter dem Dokumententitel `konstanten.m` gefunden werden.

Abbildung 6 zeigt die erfolgreiche Testung der Routine Anhand des Vergleiches zwischen der analytischen Lösung und der der Lösung über die implementierte Routine. Anhand des unteren Abschnittes der Grafik kann festgehalten werden, dass die Lösung über die Routine der realen Lösung annähernd entspricht. Der größe Fehler tritt an den Randstellen auf.

Weiterhin zeigt Abbildung 7, dass der absolute Fehler der Methode mit steigender Anzahl an Teilintervallen N abnimmt. Außerdem kann abgelesen werden, dass es sich um ein Verfahren zweiter Ordnung handelt, da der Fehler um zwei Dekaden sinkt, wenn die Anzahl der Teilintervalle um eine Dekade erhöht wird.

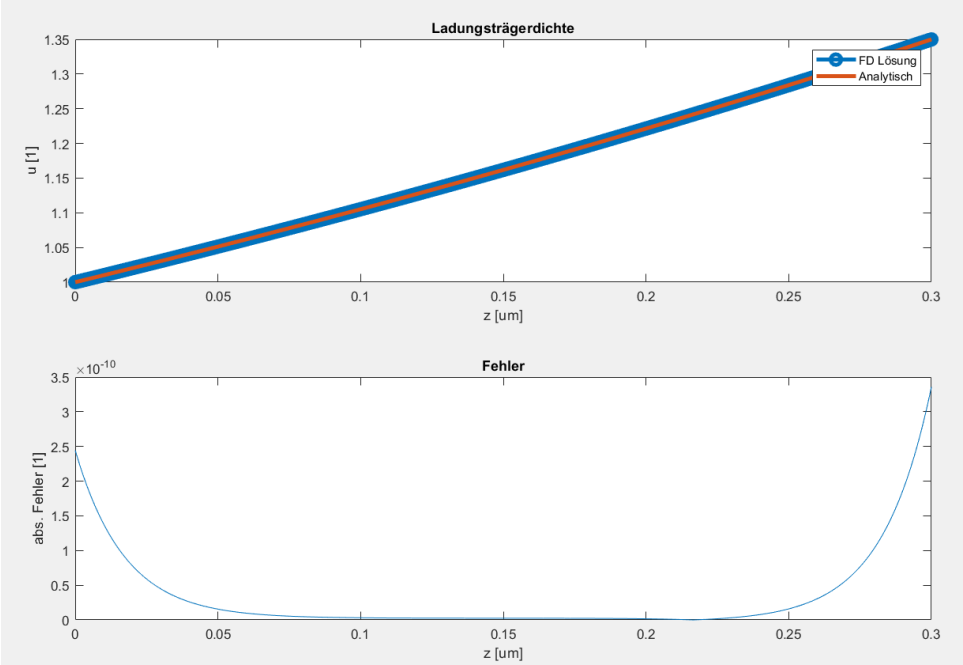

Abb. 6: Vergleich der Analytischen Lösung mit der Lösung über die entwickelte Routine, sowie Darstellung des Fehlers der Lösung mittels der Routine

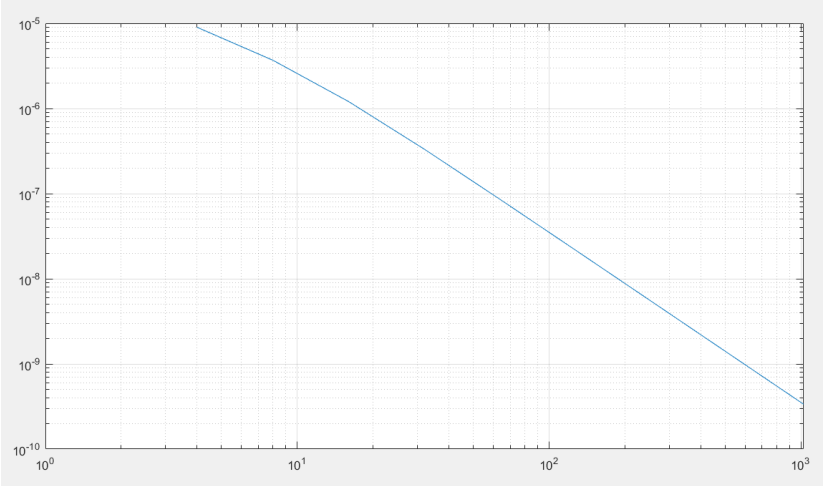

Abb. 7: Darstellung zur Ermittlung der Ordnung der Methode

Die genutzte Matlab Routine zum Testen der Routine kann unter dem Dokumentennamen `ordnung_stationär_nonlin.m` gefunden werden im beigefügten Ordner.

2.2.7 Anwendung der Routine auf spezielle Fälle

In dieser Teilaufgabe soll mit der entwickelten Routine die Lösung für die Fälle

$$s(z) = S_0 \cdot e^{-\alpha z}, \quad S_0 = 10^2, 10^3, 10^4 \frac{1}{\mu m^3 \mu s}$$

berechnet werden. Weiterhin soll ein geeignetes N so bestimmt werden, dass der relative Fehler in u_i maximal 1‰ beträgt.

Das Ergebnis für die drei Fälle von S_0 ist in Abbildung 8 zu erkennen.

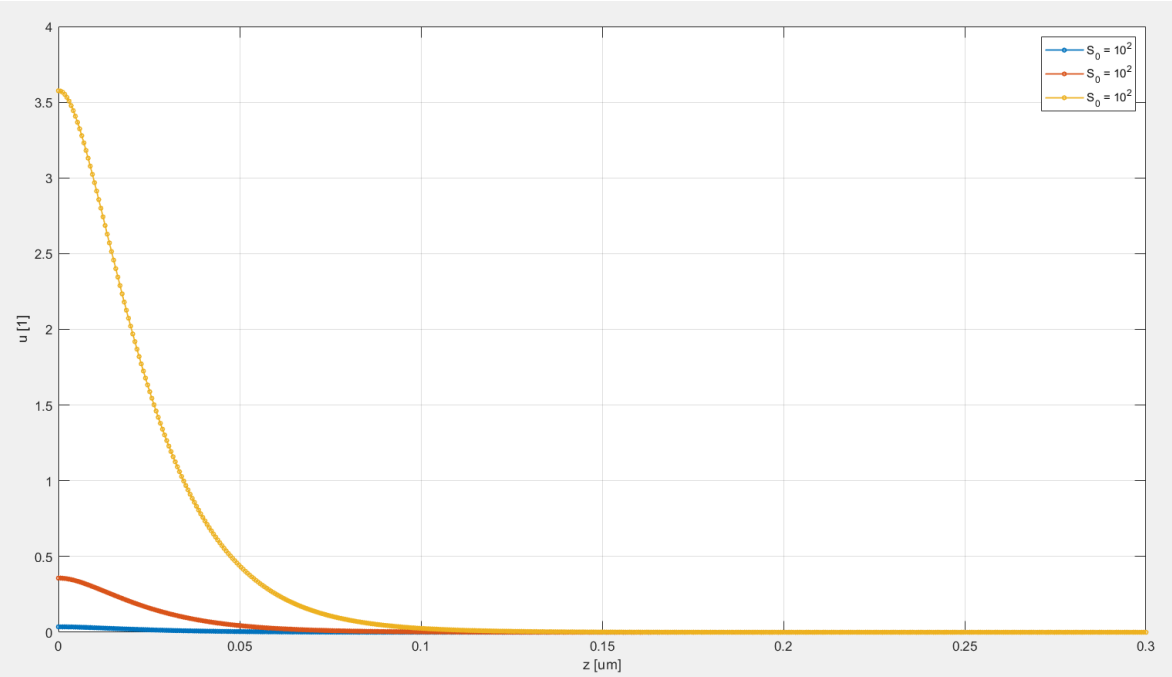

Abb. 8: Lösung von $s(z)$ für verschiedene S_0

Um ein geeignetes N zu bestimmen, um den realtiven Fehler zu erreichen, wird das N in einer Schleife iterativ erhöht. Dabei wird der Fehler relativ zum vorherigen N berechnet. Das ergebnis dieser Routine wird in Abbildung 9 dargestellt.
Die Routine kann unter dem Dokumentennamen `exper_stationär_nonlin.m` gefunden werden im beigefügten Ordner. Es kann festgehalten werden, dass das N als 512 gewählt werden muss, um einen relativen Fehler in u_i von maximal 1‰ zu erreichen.

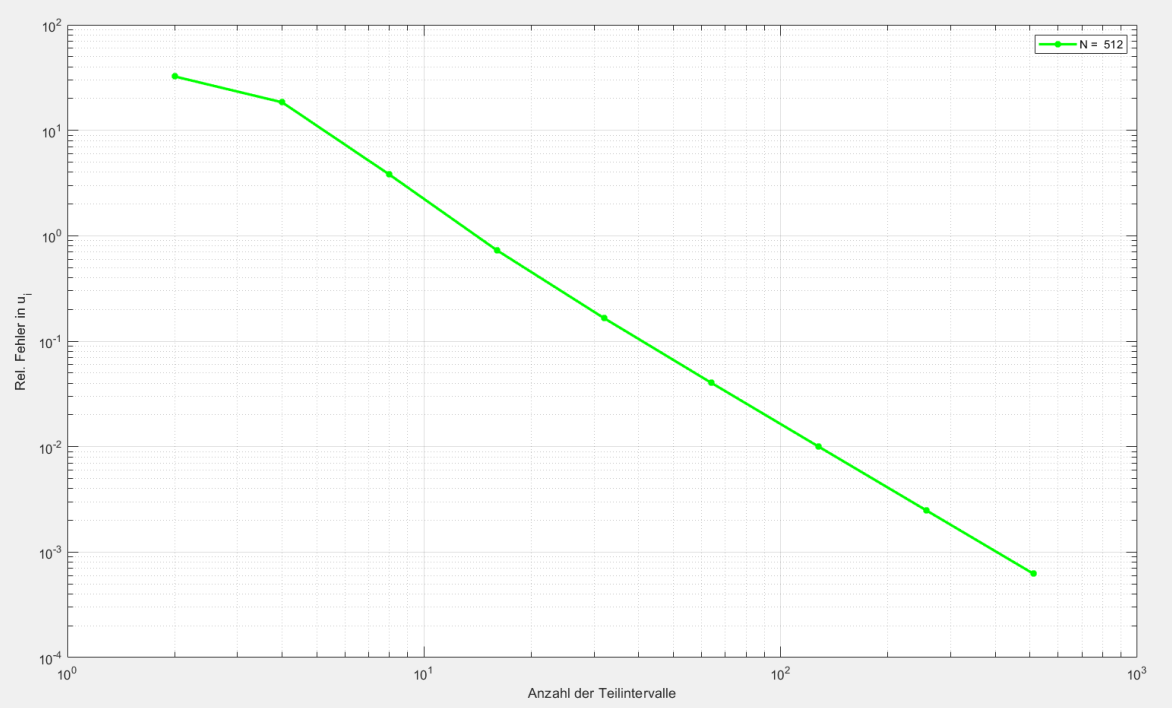

Abb. 9: Experimentelle Ermittlung von N, so dass der relative Fehler in u_i maximal 10^{-3} beträgt

Beim Vergleich von Abbildung 4 und Abbildung 8 kann festgestellt werden, dass die Lösung für alle Fälle gleich scheinen. Auch die experimentelle Bestimmung eines geeigneten N ist gibt sowohl für den linearen, als auch für den nichtlinearen Fall den selben Wert ($N = 512$) zurück.

3 Implizite Einschrittverfahren

In diesem Abschnitt sollen Löser für Systeme von sogenannten steifen Anfangswertproblemen untersucht werden. In Abschnitt 4 wird die partielle Differenzialgleichung Gleichung 1 mithilfe der finiten Differenzen des letzten Abschnitts näherungsweise als ein System von Anfangswertproblemen formuliert, das sich steif verhält.

Steife Systeme treten aber auch in der Modellierung von Reaktionen in der Chemie auf. Als Beispiel soll eine chemische Reaktion dreier Stoffe A, B, C aus [5] dienen:

$$A \to B, \quad B + B \to C + B, \quad B + C \to A + C$$

Die Dynamik der Konzentrationen der einzelnen Komponenten können durch Anfangswertproblem

$$\begin{array}{lrcrcl} A: & y_1' & = & -0.04y_1 & +10^4 y_2 y_3 & \\ B: & y_2' & = & +0.04y_1 & -10^4 y_2 y_3 & -3 \cdot 10^7 y_2^2 \\ C: & y_3' & = & & & +3 \cdot 10^7 y_2^2 \end{array} \tag{9}$$

mit den Anfangswerten

$$y_1(0) = 1, \quad y_2(0) = 0, \quad y_3(0) = 0 \tag{10}$$

modelliert werden. Anhand dieses Systems sollen im folgenden steife Differenzialgleichungen untersucht und ein effizientes numerisches Verfahren entwickelt werden.

3.1 Entwicklung

Ein allgemeines System von Anfangswertproblemen für eine Funktion $y : \mathbb{R} \to \mathbb{R}^k$ kann in der Form

$$y' = f(t, y), \quad t \in [a, b], \quad y(a) = y_a \tag{11}$$

geschrieben werden. Wie in der Vorlesung wird zur näherungsweisen numerischen Lösung das Intervall $[a, b]$ in n gleich große Teilintervalle der Länge h aufgeteilt mit Knoten $a = t_0 < t_1 < ... < t_n = b$. Der Wert von y wird näherungsweise an diesen Zeitpunkten berechnet:

$$y^{(i)} \approx y(t_i), \quad i = 0, 1, ..., N.$$

Für die Berechnung der Werte $y(i)$ sollen implizite Einschrittverfahren wie in [3, Abschnitt 8.4] verwendet werden. Insbesondere das implizite Euler-Verfahren [3, Gleichung (8.51)]

$$y^{(i+1)} = y^{(i)} + hf(t_{i+1}, y^{(i+1)}) \tag{12}$$

und die implizite Trapezregel [3, Gleichung (8.57)]

$$y^{(i+1)} = y^{(i)} + \frac{h}{2}\left[f(t_i, y^{(i)}) + f(t_{i+1}, y^{(i+1)})\right] \tag{13}$$

Im Unterschied zu [3] soll die Lösung der auftretenden nichtlinearen Gleichungssystemen nicht mithilfe der Fixpunktiteration sondern mit dem Newton-Verfahren aus der Belegarbeit näherungsweise bestimmt werden.

Dazu wird wie folgt vorgegangen.

3.1.1 Steife Differenzialgleichungen

Unter steifen Differenzialgleichungen versteht man Differenzialgleichungen, für welche bestimmte numerische Methoden zum Lösen der Gleichung instabil sind, wenn die Schrittweite nicht extrem klein gewählt wird. Es existieren Differnzialgleichungen, welche Terme besitzen, die zu starken Variationen in der Lösung führen.

3.1.2 Implizite Verfahren als Nullstellenproblem

In den Gleichungen des impliziten Euler-Verfahrens und der impliziten Trapezregel (Gleichung 12 und Gleichung 13) wird

$$y^{(i+1)} = y^{(i)} + z$$

gesetzt. Wir erhalten die folgenden Gleichungen für die beiden Verfahren:

$$\begin{aligned} Euler: &\quad y^{(i)} + z = y^{(i)} + hf(t_{i+1}, y^{(i)} + z) \\ Trapez: &\quad y^{(i)} + z = y^{(i)} + \frac{h}{2}\left[f(t_i, y^{(i)}) + f(t_{i+1}, y^{(i)} + z)\right] \end{aligned}$$

Anschließend wird jeweils die Gleichung für z als Nullstellenproblem formuliert:

$$F_{euler}(z) = 0, \quad F_{trapez}(z) = 0$$

Es ergeben sich die folgenden Nullstellenprobleme:

$$F_{euler}(z) = z - hf(t_{i+1}, y^{(i)} + z) = 0$$
$$F_{trapez}(z) = z - \frac{h}{2}\left[f(t_i, y^{(i)}) + f(t_{i+1}, y^{(i)} + z)\right] = 0$$

3.1.3 Jacobi-Matrizen DF_{euler} und DF_{trapez}

Für beide Verfahren soll aus den Gleichungen $F_{euler}(z)$ und $F_{trapez}(z)$ die Jacobi-Matrizen DF_{euler} und DF_{trapez} bestimmt werden in Abhängigkeit von $D_y f$ (der Jacobi-Matrix von f bezüglich y).

Für das implizite Euler-Verfahren ergibt sich:

$$DF_{euler}(z) = D \cdot z - D_y f(t_{i+1}, y^{(i)} + z)$$

Für die implizite Trapezregel ergibt sich:

$$DF_{trapez}(z) = \frac{h}{2}\left[D_y f(t_i, y^{(i)}) + h \cdot D_y f(t_{i+1}, y^{(i)} + z)\right] - D \cdot z$$

Dabei gilt:

$$Df = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \cdots & \frac{\partial f_2}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial y_1} & \frac{\partial f_n}{\partial y_2} & \cdots & \frac{\partial f_n}{\partial y_n} \end{bmatrix}$$

Weiterhin gilt auch:

$$Df = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

3.1.4 Matlab Routinenen zur Umsetzung der letzten beiden Teilaufgaben

Die Routinen können unter den Dokumentennamen `F_euler.m` und `F_trapez.m` gefunden werden im beigefügten Ordner.

3.1.5 Löser zu den beiden Verfahren mit Matlab

Die Löser können unter den Dokumentennamen `impl_euler.m` und `impl_trapez.m` gefunden werden im beigefügten Ordner.

3.1.6 Ermittlung der Ordnung der Verfahren

Die Ermittlung der Ordnung erfolgt anhand des Beispiels

$$y' = -y$$

mit der Anfangsbedingung

$$y(0) = 1$$

auf dem Intervall

$$t_{span} = [0, 1].$$

Zunächst wird Differenzialgleichung und deren Ableitung aufgestellt:

$$f(t, y) = -y$$
$$f'(t, y) = -1$$

Weiterhin ist die exakte Lösung bekannt:

$$y = e^{-t}$$

In einer Schleife kann nun die Schrittweite iterativ erhöht werden, um die Ordnung der beiden Verfahren grafisch zu ermitteln. Abbildung 10 und Abbildung 11 zeigen die Ergebnisse. Es ist zu erkennen, dass das implizite Euler-Verfahren eine Ordnung von 1 hat und die implizite Trapezregel eine Ordnung von 2.

Die Matlab-Scripte können unte den Dokumentennamen `ordnung_impl_euler.m` und `ordnung_impl_trapez.m` gefunden werden im beigefügten Ordner.

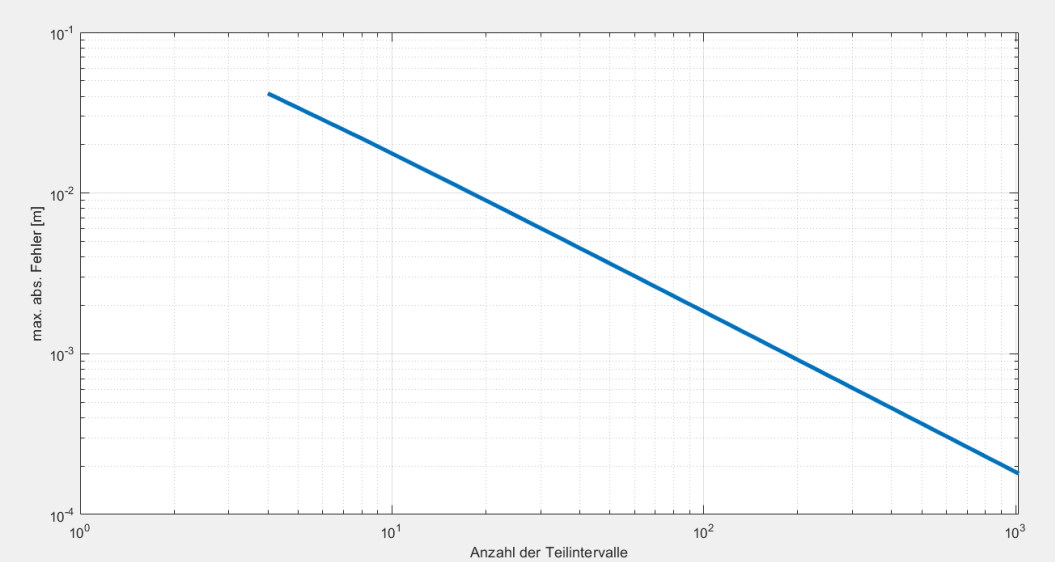

Abb. 10: Grafische Bestimmung der Ordnung des impliziten Euler-Verfahrens

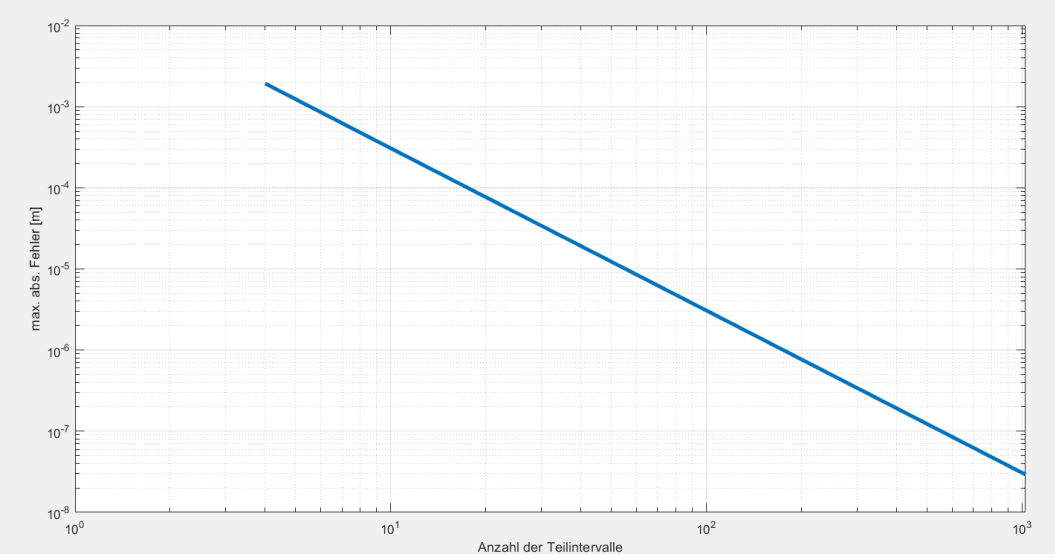

Abb. 11: Grafische Bestimmung der Ordnung der impliziten Trapezregel

3.1.7 Test der Routine

Zum Testen der Routine wird eine Lösung und die ersten beiden Ableitungen dieser vorgegeben.

$$\begin{aligned} y(t) &= e^{-t} \cdot sin(t) \\ y'(t) &= e^{-t} \cdot cos(t) - e^{-t} \cdot sin(t) \\ y''(t) &= -2 \cdot e^{-t} \cdot cos(t) \end{aligned}$$

Nachfolgend wird eine Differenzialgleichung aufgestellt, die ein System nichtlinearer, zeitabhängiger Anfangswertprobleme beschreibt:

$$r = y'' + t \cdot y + \omega_0^2 \cdot (y - \frac{1}{6} y^3)$$

Nun muss eine rechte Seite ermittelt werden, die für die vorgegebene Lösung passend ist. Dazu wird zunächst ein Spaltenvektor aus der Lösung und deren 1. Ableitung gebildet als Funktionshandle abhängig von t und z.

$$z(t, z) = \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix}$$

Nachfolgend wird die Ableitung dieses Vektors bestimmt. Die erste Zeile enthält nun die 1. Ableitung von y. Die zweite Zeile enthält die 2. Ableitung von y. Durch das Umstellen der Differenzialgleichung r nach y'' kann nun wie folgt eingesetzt werden:

$$f_z(t, z) = \begin{bmatrix} y'(t) \\ r(t) - \left(t \cdot y'(t) + \omega_0^2 \cdot \left(y(t) - \frac{1}{6} \cdot y(t)^3 \right) \right) \end{bmatrix}$$

Da das Newton-Verfahren auch die Ableitung dieses Vektors benötigt, muss im nächsten Schritt die Jacobi-Matrix berechnet werden.

$$Df_z(t, z) = \begin{bmatrix} 0 & 1 \\ -\omega_0 + \omega_0 \cdot 3 \cdot z_1^2 & -t \end{bmatrix}$$

Abschließend wird der Startwert ermittelt.

$$z_0 = \begin{bmatrix} y(0) \\ y'(0) \end{bmatrix}$$

Nun können die berechneten Gleichungen als Parameter in der Matlab Routine für die implizite Trapezregel eingesetzt werden. Das Ergebnis ist in Abbildung 12 dargestellt.

Das Matlab-Sckript mit dem Test der Routine kann unter dem Dokumentennamen `test_routine.m` gefunden werden im beigefügten Ordner.

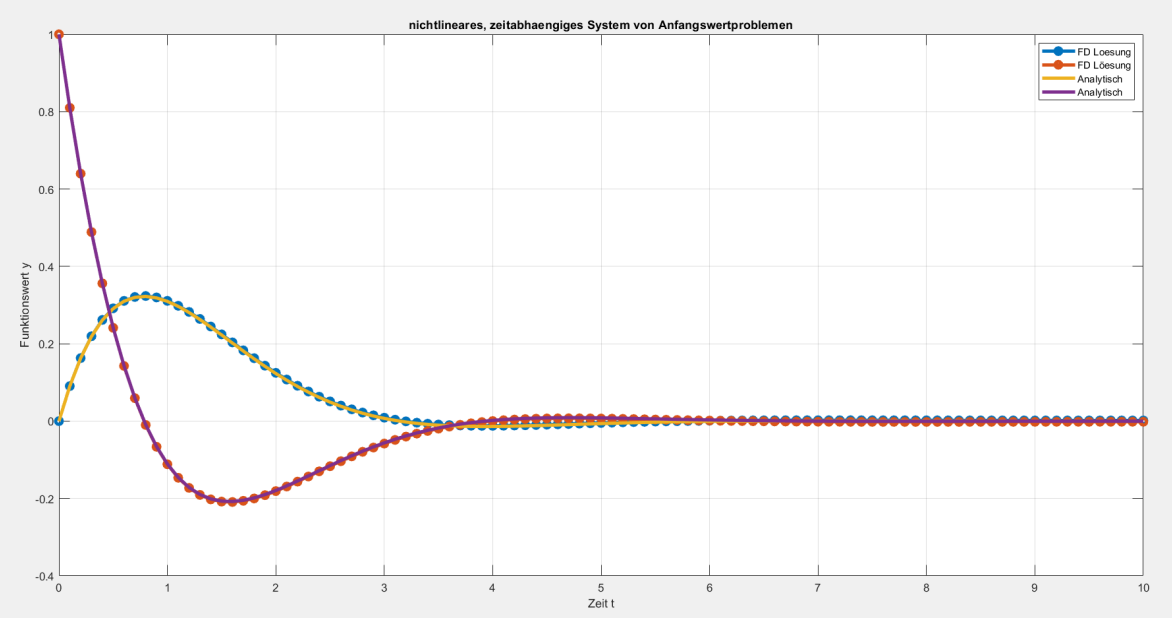

Abb. 12: Test der Routine mit einem nichtlinearen, zeitabhängigen System von Anfangswertproblemen

3.2 Anwendung

In diesem Abschnitt sollen die entwickelten Löser auf das Modell der chemischen Reaktion angewandt werden. Dabei wird wie folgt vorgegangen:

3.2.1 Gleichung 9 als System der Form (11) mit Funktion $f_{chem}(t, y)$

$$y' = f_{chem}(t, y) = \begin{bmatrix} -0.04y_1 + 10^4 y_2 y_3 \\ 0.04y_1 - 10^4 y_2 y_3 - 3 \cdot 10^7 y_2^2 \\ 3 \cdot 10^7 y_2^2 \end{bmatrix}$$
$$mit \quad t \in [0, \backsim]$$
$$y(a) = y_a = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

3.2.2 Matlab Routine für die chemische Reaktion

Die implementierte Matlab Routine kann unter dem Dokumentennamen `f_chem.m` gefunden werden im beigefügten Ordner.

3.2.3 Jacobi-Matrix $Dyf_{chem}(t, y)$ von f_{chem} bezüglich y

$$Dyf_{chem}(t, y) = \begin{bmatrix} -0.04 & 10^4 y_3 & 10^4 y_2 \\ 0.04 & -10^4 y_3 - 6 \cdot 10^7 y_2 & -10^4 y_2 \\ 0 & 6 \cdot 10^7 y_2 & 0 \end{bmatrix}$$

3.2.4 Jacobi-Matrix $Df_{chem}(t, y)$ als Matlab Routine

Die implementierte Matlab Routine kann unter dem Dokumentennamen `f_chem_jac.m` gefunden werden im beigefügten Ordner.

3.2.5 Lösung des Anfangswertproblemes mit verschiedenen Integratoren

Abbildung 13 zeigt Beispielhaft die Lösung des Anfangswertproblemes auf dem Intervall $[0, 1]$ mit der impliziten Trapezregel.

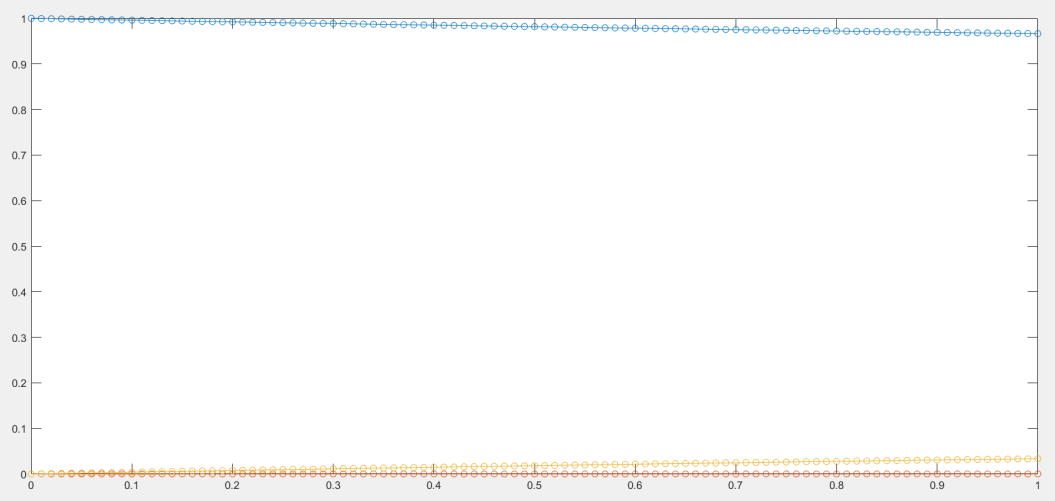

Abb. 13: Grafische Darstellung der Lösung des Anfangswertproblemes mit den verschiedenen Integratoren am Beispiel der impliziten Trapezregel

Mit der Matlab Routine unter dem Dokumentennamen `exper_chem.m` wurde experimentell für die vier umgesetzten Integratoren (Expliziter Euler, Mitelpunktsregel, Impliziter Euler, Implizite Trapezregel) ein geeignetes n so bestimmt, dass der absolute Fehler jeder einzelnen Komponente und Zeitschritts höchstens 10^{-4} bzw. 10^{-6} beträgt.
Tabelle 2 und Tabelle 3 zeigen die Ergebnisse der experimentellen Berechnungen.

Verfahren	n	abs. Fehler in y_1	abs. Fehler in y_2	abs. Fehler in y_3
Expliziter Euler	2048	$0.1191 \cdot 10^{-5}$	$0.0000 \cdot 10^{-5}$	$0.1191 \cdot 10^{-5}$
Mittelpunktsregel	2048	$0.2961 \cdot 10^{-5}$	$0.3075 \cdot 10^{-5}$	$0.0114 \cdot 10^{-5}$
Impliziter Euler	32	$0.7404 \cdot 10^{-4}$	$0.0001 \cdot 10^{-4}$	$0.7406 \cdot 10^{-4}$
Implizite Trapezregel	32	$0.7960 \cdot 10^{-4}$	$0.0263 \cdot 10^{-4}$	$0.7707 \cdot 10^{-4}$

Tab. 2: Experimentelle Bestimmung von n bei einem maximalen Fehler von 10^{-4}

Verfahren	n	abs. Fehler in y_1	abs. Fehler in y_2	abs. Fehler in y_3
Expliziter Euler	8192	$0.2978 \cdot 10^{-6}$	$0.0000 \cdot 10^{-6}$	$0.2978 \cdot 10^{-6}$
Mittelpunktsregel	16384	$0.3852 \cdot 10^{-6}$	$0.4262 \cdot 10^{-6}$	$0.0411 \cdot 10^{-6}$
Impliziter Euler	4096	$0.5954 \cdot 10^{-6}$	$0.0001 \cdot 10^{-6}$	$0.5955 \cdot 10^{-6}$
Implizite Trapezregel	1024	$0.4012 \cdot 10^{-9}$	$0.0001 \cdot 10^{-9}$	$0.4012 \cdot 10^{-9}$

Tab. 3: Experimentelle Bestimmung von n bei einem maximalen Fehler von 10^{-6}

3.2.6 Diskussion der Performanz

Vor allem an Tabelle 3 it gut zu erkennen, dass das explizite Euler-Verfahren ein kleineres n benötigt, um einen vergleichbaren Fehler zur Mittelpunktsregel zu erreichen.
Ähnliches kann auch im Vegleich der beiden impliziten Verfahren festgestellt werden. Die implizite Trapezregel benötigt ein kleineres n um einen vergleichbaren Fehler zum impliziten Euler-Verfahren zu erreichen.
Weiterhin kann festgehalten werden, dass die impliziten Verfahren ein deutlich geringeres n für einen vergleichbaren Fehler benötigen. Daraus kann für die Performanz der Routinen die Aussage getroffen werden, dass die impliziten Verfahren eine bessere Performanz haben als die expliziten Verfahren.

Die vier implementierten Löser können unter den Dokumentennamen `euler_systeme.m`, `mittelpunktsregel_systeme.m`, `impl_euler.m` und `impl_trapezregel.m` im beigefügten Ordner gefunden werden. Das Matlab File mit der experimentellen Ermittlung des n ist unter dem Dokumentennamen `exper_chem.m` zu finden.

4 Zeitaufgelöste Simulation

In diesem Abschnitt werden die entwickelten Methoden kombiniert, um die ursprüngliche Gleichung 1 zu lösen. Dazu wird diese Gleichung an den Knotenpunkten $z_0, ..., z_N$ aus Abschnitt 2 ausgewertet und man erhält

$$\frac{\partial u}{\partial t}(t, z_i) = D\frac{\partial^2 u}{\partial z^2}(t, z_i) - (k_1 + k_2 N_D)u(t, z_i) - k_2 u^2(t, z_i) + s(t, z_i), \quad t \geq t_0, i = 0, 1, ..., N$$

Wie bei den finiten Differenzen setzen wir

$$u_i(t) = u(t, z_i)$$

und approximieren die rechte Seite durch finite Differenzen wie in Abschnitt 2. Wir erhalten somit ein System von Anfangswertproblemen für die Funktionen $u_i(t), \quad i = 0, ..., N$. Es wird wie folgt vorgegangen.

4.1 Formulierung der Differenzialgleichung als System

Das System soll in der Form

$$u' = F(t, u) \tag{14}$$

dargestellt werden. Wie auch schon in Abschnitt 2 werden die Randbedingungen aus Gleichung 5 berücksichtigt.

weiterhin gilt:

$$\begin{aligned} k &= k_1 + k_2 \cdot N_D \\ s_i &= s(z_i) \\ h &= \frac{d}{N} \\ u_i &\approx u(i) \end{aligned}$$

Anolog werden die 1. und 2. Ableitung approximiert zu:

$$u' = \frac{u_{i+1} - u_{i-1}}{2h}, \quad u'' = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}$$

Durch das Einsetzen der Approximationen der beiden Ableitungen ergeben sich für u_i mit $i = 0, ..., N$ unter der der Festlegung $s_i = s(z_i)$ und $u_i \approx u(z_i)$ folgende Gleichungen:

$$u_0' = D \cdot \frac{u_1 - 2 \cdot u_0 + u_{-1}}{h^2} - k \cdot u_0 - k_2 \cdot u_0^2 + s_0 \quad mit \quad i = 0$$
$$u_i' = D \cdot \frac{u_{i+1} - 2 \cdot u_i + u_{i-1}}{h^2} - k \cdot u_i - k_2 \cdot u_0^2 + s_i$$
$$u_N' = D \cdot \frac{u_{N+1} - 2 \cdot u_N + u_{N-1}}{h^2} - k \cdot u_N - k_2 \cdot u_0^2 + s_N \quad mit \quad i = N$$

Anschließend werden die ersten Ableitungen an den Randbedingungen (Gleichung 5) approximiert durch:

$$u'(0) \approx \frac{u_1 - u_{-1}}{2h}, \quad u'(d) \approx \frac{u_{N+1} - u_{N-1}}{2h}$$

Durch das Auflösen nach u_{-1} bzw. u_{N+1} ergibt sich:

$$u_{-1} = -(u'(0) \cdot 2h) + u_1, \quad u_{N+1} = u'(d) \cdot 2h + u_{N-1}$$

Einsetzen an den Knoten z_0 und z_N ergibt:

$$u_0' = D * \frac{u_1 - 2u_0 - (u_0' \cdot 2h) + u_1}{h^2} - k \cdot u_0 - k_2 \cdot u_0^2 + s_0 \quad mit \quad i = 0$$
$$u_N' = D * \frac{u_N' \cdot 2h + u_{N-1} - 2u_N + u_{N-1}}{h^2} - k \cdot u_N - k_2 \cdot u_0^2 + s_N \quad mit \quad i = N$$

Nun können die Randbedingungen aus Gleichung 5 nach den 1. Ableitungen umgestellt und eingesetzt werden:

$$u'(0) = \frac{S_L \cdot u_0}{D}, \quad u'(N) = \frac{-S_R \cdot u_N}{D}$$

$$u_0' = D * \frac{u_1 - 2u_0 - (\frac{S_L \cdot u_0}{D} \cdot 2h) + u_1}{h^2} - k \cdot u_0 - k_2 \cdot u_0^2 + s_0 \quad mit \quad i = 0$$
$$u_N' = D * \frac{\frac{-S_R \cdot u_N}{D} \cdot 2h + u_{N-1} - 2u_N + u_{N-1}}{h^2} - k \cdot u_N - k_2 \cdot u_0^2 + s_N \quad mit \quad i = N$$

Die Gleichungen an den Stellen $i = 0, i, N$ werden nun vereinfacht bzw. umgeformt und in die Form $u' = F(t, u)$ überführt:

$$F(t,u) = \begin{bmatrix} \frac{D}{h^2} \cdot \left[2u_1 - \left(\frac{2D+S_L \cdot 2h+h^2k}{D} + \frac{h^2 \cdot k_2}{D} \cdot u_0\right) \cdot u_0\right] + s_0 \\ \vdots \\ \frac{D}{h^2} \cdot \left[u_{i+1} - \left(\frac{2D+h^2k}{D} + \frac{h^2 \cdot k_2}{D} \cdot u_i\right) \cdot u_i + u_{i-1}\right] + s_i \\ \vdots \\ \frac{D}{h^2} \cdot \left[\left(\frac{-S_R \cdot 2h-2D-h^2k}{D} - \frac{h^2 \cdot k_2}{D} \cdot u_N\right) \cdot u_N + 2u_{N-1}\right] + s_N \end{bmatrix}$$

4.2 Matlab Routine zur Berechnung von $F(t, u)$ und $D_u F(t, u)$

Die beiden Routinen können unter den Dokumentennamen `fd_nonlin.m` und `fd_nonlin_jac.m` gefunden werden im beigefügten Ordner.

4.3 Simulation des Systems

Der Aufruf der Trapezregel für die Funktion

$$s(t,z) = S_0 \cdot e^{-\frac{t^2}{2 \cdot 0.01^2}} \cdot e^{-\alpha z}$$

kann im beigefügten Ordner unter dem Dokumententitel `error_func.m` gefunden werden. Dort wird auch der Wert für S_0 vorgegeben, sowei alle weiteren Konstatentn und Parameter.

Die Vorgabe der Zeit- und Ortsschritte erfolgt in der Matlab Routine unter dem Dokumenmtennamen `dgl_full.m`, in welcher auch die zuvor genannte Routine aufgerufen wird.

4.4 Grafische Darstellung der Ergebnisse

Abbildung 14, Abbildung 15 und Abbildung 16 zeigen die Ergebnisse der Simulation bei verschiedenen S_0. Die Ergebnisse sind in der Form einer Zeit- und Ortsplots dargestellt (Matlab `mesh`-Befehl). Es ist zu erkennen, dass je Größer das S_0 gewählt wird, umso größer ist auch die Ladungsträgerdichte zum Startzeitpunkt. Die zeitlichen und örtlichen Eigenschaften bleiben von S_0 unbeeinflusst.

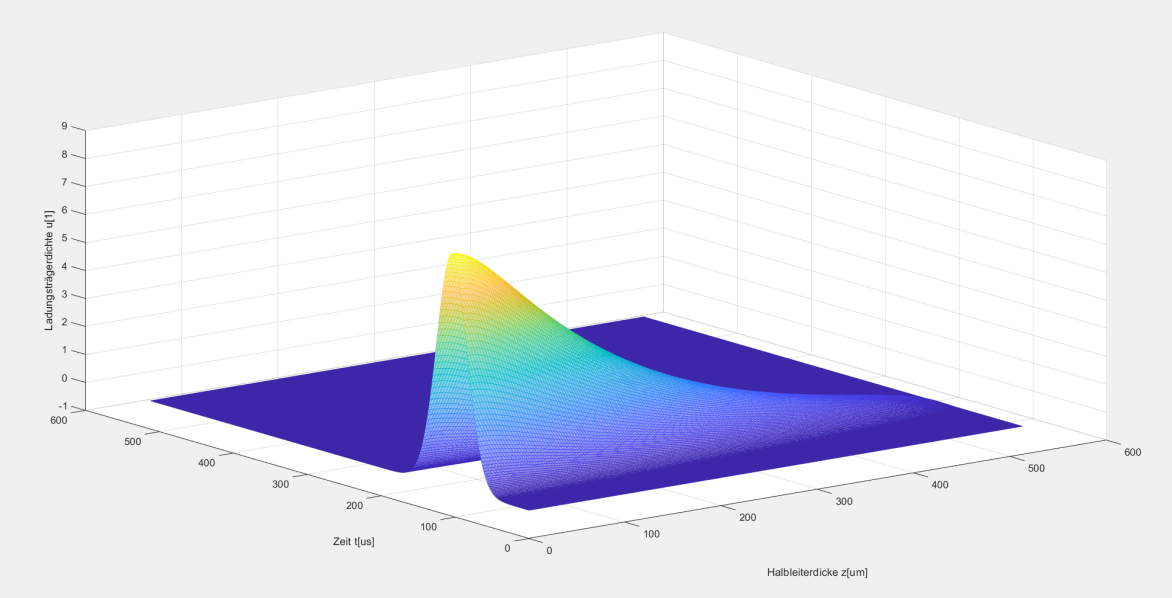

Abb. 14: Grafische Darstellung der Zeitaufgelösten Simulation bei $S_0 = 10^4$

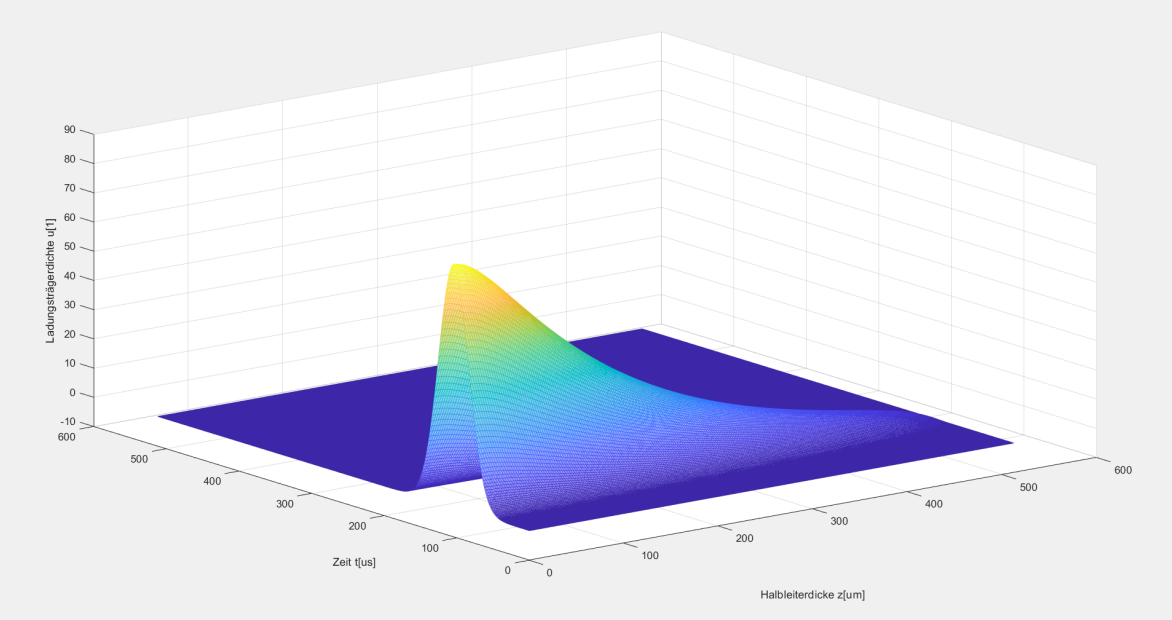

Abb. 15: Grafische Darstellung der Zeitaufgelösten Simulation bei $S_0 = 10^5$

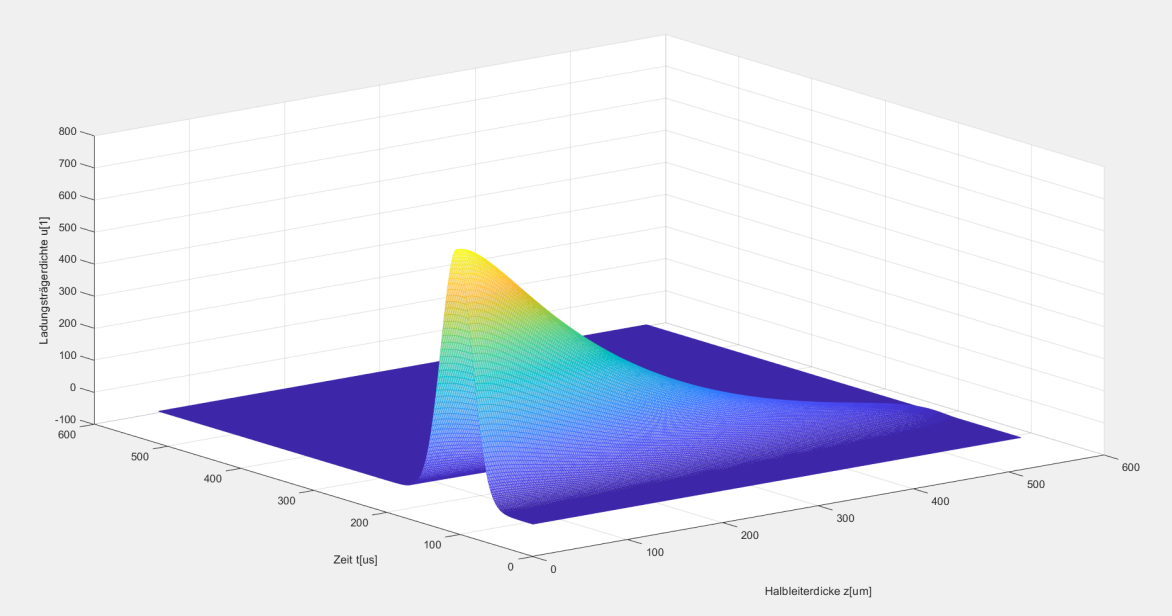

Abb. 16: Grafische Darstellung der Zeitaufgelösten Simulation bei $S_0 = 10^6$

Literaturverzeichnis

[1] HTW-Logo auf dem Deckblatt
https://de.wikipedia.org/wiki/Datei:Logo_HTW_Berlin.svg
Stand: 17.08.2018 um 14:49 Uhr

[2] HTW-Logo in der Kopfzeile
http://tonkollektiv-htw.de/
Stand: 17.08.2018 um 14:53 Uhr

[3] Baloch, Ahmer A. B. *et al:* „Analysis of Photocarrier Dynamics at Interfaces in Pervoskite Solar Cells by Time-Resolved Photoluminescensce“, The Journal of Physical Chemistry C, Seiten 26805 - 26815 (2018).

[4] Atkinson, Kendall E. und Han, Weimin: „Elementary Numerical Analysis“, J. Wiley & Sons, Hoboken, NJ (2004).

[5] Hairer, Ernst und Wanner, Gerhard: „Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems“, Springer, Berlin (1991).