



Projekt Zeitaufgelöste Photolumineszenz

Hausarbeit

im Studienfach
Angewandte Mathematik

an der

Hochschule für Technik und Wirtschaft Berlin
Fachbereich I Energie und Information
Studiengang Elektrotechnik

1. Prüfer: Max Mustermann
2. Prüfer: Max Mustermann

Eingereicht von: Milan Daniel Larsen
Matrikelnummer: s0000000
Datum der Abgabe: 25.04.2017

Inhaltsverzeichnis

1 Einleitung	1
2 Finite Differenzen der stationären Gleichung	2
2.1 Lineare stationäre Gleichung	3
2.2 Nicht Lineare stationäre Gleichung	17
3 Implizite Einschrittverfahren	26
3.1 Entwicklung	27
3.2 Anwendung	36
4 Zeitaufgelöste Simulation	42
Abbildungsverzeichnis	A
Literatur	B

1 Einleitung

In dieser Hausarbeit sollen die Grundlagen einer Simulation der Dynamik in neuartigen Perowskit-Solarzellen gelegt werden. Diese Art der Dünnschicht Solarzellen erreicht hohe Wirkungsgerade von über 20% und ist somit für die Forschung von großer Interesse[1].

2 Finite Differenzen der stationären Gleichung

Im folgendem Kapitel soll die stationäre Verteilung der Ladungsträger bei kontinuierlicher Bestrahlung modelliert werden. Dadurch kann die zeitliche Abhängigkeit vernachlässigt werden ($\frac{\partial u}{\partial t} = 0$)

Die allgemeine DGL ist gegeben durch:

$$\frac{\partial u}{\partial t} = D \cdot \frac{\partial^2 u}{\partial z^2} - (k_1 + k_2 \cdot N_D) \cdot u - k_2 u^2 + s(t, z) \quad (2.1)$$

mit: $s(t, z)$ = Ladungsträgerdichte durch Bestrahlung

D = Diffusionskonstante

N_D = Dotierungsichte

k_1, k_2 = Rekombinationskonstanten

α = Absorptionskonstante

S_L, S_R = Rekombinationsraten an den jeweiligen Grenzschichten

Mit $\frac{\partial u}{\partial t} = 0$ folgt die stationäre Gleichung:

$$D \cdot \frac{\partial^2 u}{\partial z^2} - (k_1 + k_2 N_D) \cdot u - k_2 \cdot u^2 = -s(z), \quad 0 < z < d \quad (2.2)$$

mit den Randbedingungen:

$$D \cdot \frac{\partial u}{\partial z}(0) = S_L u(0), \quad D \frac{\partial u}{\partial z}(d) = -S_R u(d) \quad (2.3)$$

2.1 Lineare stationäre Gleichung

Im Folgendem Kapitel soll nur der in u lineare Anteil der stationären, zeitunabhängigen Gleichung (Eq. 2.2) ohne den quadratischen Term $-k_2u^2$ behandelt werden[1].

Aufgabe 1. Erarbeiten Sie sich Abschnitt 8.8 aus [2] und beschreiben Sie Ihr Vorgehen für die Anwendung der Methode auf Gleichung.[1]

Mit der Methode der finiten Differenzen für Zweipunkt-Grenzwertprobleme, beschrieben in [2, p. 442], können lineare Differentialgleichungen zweiter Ordnung gelöst werden. Es werden dabei die Ableitungen der Gleichung durch finite Differenzen substituiert, um eine diskretisierte Gleichung zu erhalten. Als Ergebnis erhält man eine Matrix, mit der sich die zeitabhängige Stelle der Leitungsträgerdichte u_i mathematisch beschreiben lassen.

Für diese Methode sind drei Schritte notwendig:

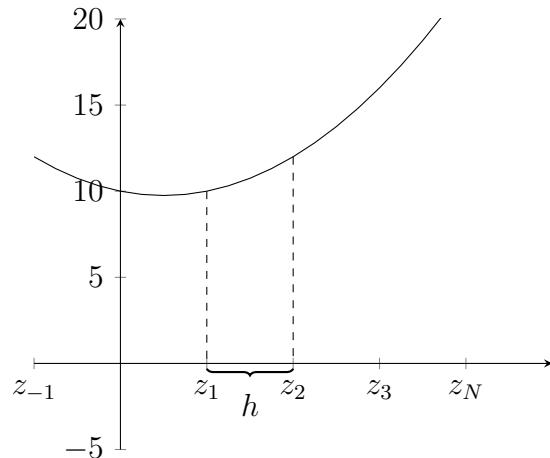
1. Diskretisierung eines Bereiches ($z \in [0, d]$) der Funktion in M gleich große Intervalle der Länge $h = \frac{(d-0)}{N}$ (Abb. 2.1)
2. Diskretisierung der Differentialgleichung an den Knotenpunkten mit Approximation der Ableitung (Gleichungen (2.5) und (2.6))
3. Diskretisierung der Randbedingung und Aufstellung eines Gleichungssystems durch Eliminierung von u_{-1} und u_{N+1}

Aufgabe 2. Leiten Sie analog zu Gleichung (8.128) die Gleichungen für die gesuchten Werte u_0, u_1, \dots, u_N an den Punkten z_0, \dots, z_N her. Die Gleichungen enthalten u_1 und u_{N+1} , die im nächsten Schritt eliminiert werden. Verwenden Sie dabei die Abkürzung $s_i = s(z_i)$.[1]

Stationäre Differentialgleichung(Gleichung (2.2)) ohne den quadratischen Term $-k^2u^2$:

$$D \frac{\partial^2 u}{\partial z^2} - ku = -s(z), \quad 0 < z < d \quad (2.4)$$

mit $k = k_1 + k_2 \cdot N_D$.

Abbildung 2.1: Beispieldarstellung: Knotenpunkte mit Schrittweite h

$$k = k_1 + k_2 * N_D$$

$$s_i = s(z_i)$$

$$u_i \approx u(z_i)$$

mit: u_i = Approximierte Ladungsträgerdichte an den Stellen z_i

s_i = Ladungsträgerdichte die durch externe Bestrahlung an den Stellen z_i erzeugt wird

Approximation der Ableitung erster Ordnung:

$$\frac{\partial u}{\partial z} = \frac{u_{i+1} - u_{i-1}}{2h} \quad (2.5)$$

Approximation der Ableitung zweiter Ordnung:

$$\frac{\partial^2 u}{\partial z^2} = \frac{u_{i+1} - 2 \cdot u_i + u_{i-1}}{h^2} \quad i = 1, 2, \dots, N \quad (2.6)$$

Durch Einsetzen von Gleichung (2.6) in Gleichung (2.4) erhält man:

$$D \cdot \frac{u_{i+1} - 2 \cdot u_i + u_{i-1}}{h^2} - k \cdot u_i = -s_i \quad (2.7)$$

Nun wird nach der Variable u_i sortiert und folgende Gleichung aufgestellt:

$$\frac{D}{h^2} \cdot u_{i-1} - \frac{2D + kh^2}{h^2} \cdot u_i + \frac{D}{h^2} \cdot u_{i+1} = -s_i \quad (2.8)$$

Aufgabe 3. Approximieren Sie die ersten Ableitungen an den Randbedingungen:

$$D \cdot \frac{\partial u}{\partial z}(0) = S_L u(0), \quad D \frac{\partial u}{\partial z}(d) = -S_R u(d) \quad (2.9)$$

durch:

$$u'(0) \approx \frac{u_1 - u_{-1}}{2h}, \quad u'(d) \approx \frac{u_{N+1} - u_{N-1}}{2h} \quad (2.10)$$

und lösen Sie die Gleichungen nach u_1 bzw. u_{N+1} auf. Setzen Sie diese Ausdrücke in die Gleichungen für die Knoten z_0 und z_N der letzten Teilaufgabe ein.

Erste Randbedingung:

$$D \cdot \frac{\partial u}{\partial z}(0) = S_L u(0) \quad (2.11)$$

$$\frac{\partial u(0)}{\partial z} = \frac{S_L u(0)}{D} \quad (2.12)$$

Mit der Approximation der Ableitung erster Ordnung (Gleichung (2.5)):

$$\frac{S_L u_0}{D} = \frac{u_1 - u_{-1}}{2h} \quad (2.13)$$

$$u_0 = D \cdot \frac{u_1 - u_{-1}}{2h \cdot S_L} \quad (2.14)$$

$$u_0 = \frac{D}{2h \cdot S_L} \cdot u_1 - \frac{D}{2h \cdot S_L} \cdot u_{-1} \quad (2.15)$$

Damit folgt für u_{-1} :

$$u_{-1} = u_1 - \frac{2h \cdot S_L}{D} \cdot u_0 \quad (2.16)$$

Zweite Randbedingungen:

$$D \frac{\partial u}{\partial z}(d) = -S_R u(d) \quad (2.17)$$

$$\frac{\partial u}{\partial z}(d) = -S_R u(d)/D \quad (2.18)$$

Durch Approximation der Randbedingung folgt:

$$\frac{-S_R}{D} \cdot u_N = \frac{u_{N+1} - u_{N-1}}{2h} \quad (2.19)$$

$$u_N = D \cdot \frac{u_{N+1} - u_{N-1}}{-2hS_R} \quad (2.20)$$

$$u_N = \frac{D}{-2hS_R} \cdot u_{N+1} - \frac{D}{-2hS_R} \cdot u_{N-1} \quad (2.21)$$

Damit folgt für u_{N+1} :

$$u_{N+1} = u_{N-1} - \frac{2 \cdot h \cdot S_R}{D} \cdot u_N \quad (2.22)$$

Im folgendem werden die Approximationen der Randbedingungen in die stationäre, diskretisierte DGL (Gleichung (2.7)) eingesetzt. Dazu wird die Diskretisierung zunächst an die Randbedingungen angepasst:

$$\begin{aligned} Z_N : \quad & D \cdot \frac{u_{N+1} - 2u_N + u_{N-1}}{h^2} - ku_N = -s_N \\ Z_0 : \quad & D \cdot \frac{u_1 - 2u_0 + u_{-1}}{h^2} - ku_0 = -s_0 \end{aligned} \quad (2.23)$$

mit der Approximation der Randbedingung Gleichungen (2.16) und (2.22) folgt:

$$\begin{aligned}
 Z_N : \quad & D \cdot \frac{u_{N-1} - \frac{2 \cdot h \cdot S_R}{D} \cdot u_N - 2u_N + u_{N-1}}{h^2} - ku_N = -s_N \\
 & \frac{-2hS_R - 2D}{h^2} \cdot u_N + \frac{2D}{h^2} \cdot u_{N-1} - ku_N = -s_N \\
 & \frac{2D}{h^2} \cdot u_{N-1} - \frac{2hS_R + 2D + kh^2}{h^2} \cdot u_N = -s_N
 \end{aligned} \tag{2.24}$$

$$\begin{aligned}
 Z_0 : \quad & D \cdot \frac{u_1 - 2u_0 + u_1 - \frac{2 \cdot h \cdot S_L}{D} \cdot u_0}{h^2} - ku_0 = -s_0 \\
 & \frac{2D}{h^2} \cdot u_1 - \frac{2D + 2h \cdot S_L + kh^2}{h^2} \cdot u_0 = -S_0
 \end{aligned} \tag{2.25}$$

Es ergibt sich der Wert für Z_0 damit zu:

$$Z_0 : \quad \frac{2D}{h^2} \cdot u_1 - \frac{2D + 2h \cdot S_L + kh^2}{h^2} \cdot u_0 = -s_0 \tag{2.26}$$

Es ergibt sich der Wert für Z_N damit zu:

$$Z_N : \quad \frac{2D}{h^2} \cdot u_{N-1} - \frac{2hS_R + 2D + kh^2}{h^2} \cdot u_N = -s_N \tag{2.27}$$

Aufgabe 4. Stellen Sie das lineare Gleichungssystem für die Größen analog zu Gleichung (8.133) in [2] und folgende dar:

$$\mathbf{A}u = \mathbf{b}, \quad \mathbf{u} = [u_i], \quad \mathbf{b} = [b_i] \tag{2.28}$$

Ordnen Sie die Gleichungen analog zu den Knotenpunkten.

Mit Hilfe der Matrixform $\mathbf{A}u = \mathbf{b}$ lässt sich dieses Gleichungssystem mit Hilfe von Matlab lösen. Dazu wird die Koeffizientenmatrix

$$\mathbf{A} = \begin{bmatrix} -\frac{2hS_L+2D+kh^2}{h^2} & \frac{2D}{h^2} & & & \\ \frac{D}{h^2} & -\frac{2D+kh^2}{h^2} & \frac{D}{h^2} & & \\ \ddots & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & \ddots \\ & & \frac{D}{h^2} & -\frac{2D+kh^2}{h^2} & \frac{D}{h^2} \\ & & \frac{2D}{h^2} & -\frac{2hS_R+2D+kh^2}{h^2} & \end{bmatrix} \quad (2.29)$$

sowie die rechte Seite \mathbf{b} und der gesuchte Vektor \mathbf{u}

$$\vec{b} = -\vec{s}_i \quad , \quad \vec{u} = -\vec{u}_i \quad i = 0, 1, \dots, N \quad (2.30)$$

$$\mathbf{b} = \begin{bmatrix} -s_0 \\ -s_1 \\ \vdots \\ -s_{N-1} \\ -s_N \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} \quad (2.31)$$

bestimmt.

Aufgabe 5. : Implementieren Sie eine Routine zur Berechnung der Matrix A der letzten Teilaufgabe. Verwenden Sie dabei dünnbesetzte Matrizen (sparse matrix, siehe z.B. spdiags).

```

1 function [F] = fd_lin_matrix(N)
2 % Eingabe:
3 % N Anzahl von Teilintervallen N.
4 % Ausgabe:
5 % A Finite Differenzen Matrix für u der Größe (N + 1) x (N + 1) als ←
6 % sparse matrix.
7
8 c=konstanten; % Lade Konstanten
9 h=c.d/N;% Schrittgröße

```

```

10 % Matrix besetzen
11 A=ones(N+1,3).*[c.D/h^2 -(2*c.D+c.k*h^2)/(h^2) c.D/h^2] ;
12 F=spdiags(A, [-1, 0, 1], N+1, N+1);
13 % Randwerte in Matrix einfügen
14 F(1,1)=-(2*h*c.SL+2*c.D+c.k*h^2)/h^2;
15 F(1,2)=2*c.D/h^2;
16 F(N+1,N+1)=-(2*h*c.SR+2*c.D+c.k*h^2)/(h^2);
17 F(N+1,N)=2*c.D/h^2;
18 end

```

Aufgabe 6. : Implementieren Sie eine Routine zur Lösung des entsprechenden Gleichungssystems Gleichung (2.28) unter Verwendung `fd_lin_matrix`. Lösen Sie das lineare Gleichungssystem mithilfe des Operators \.

```

1 function [z,u] = stationaer_lin(s,N)
2 % Eingabe:
3 % s Funktionshandle auf Funktion s(z)
4 % function sz=s(z) mit Spaltenvektoren z und sz
5 % N Anzahl von Teilintervallen N.
6 % Ausgabe:
7 % z Knotenpunkte (z0; z1, . . . , zN) der Größe (N + 1) x 1
8 % u Vektor u der Größe (N + 1) x 1
9 c=konstanten;
10
11 h=c.d/N; %Schriftweite
12 z=(0:N)'*h; % Knotenpunkte
13
14 b=-s(z); % Bestimmung von b an den Knotenpunkten
15 A=fd_lin_matrix(N); % matirx berechnen
16 u=A\b;

```

Tabelle 2.1: Konstanten

Parameter	Wert	Einheit
d	0.3	μm
D	0.3	$\mu\text{m}^2/\mu\text{s}$
k	1001	μs^{-1}

Aufgabe 7. : Testen Sie Ihre Routine, indem Sie sich eine Funktion $u(z)$ und Konstanten d, D, k vorgeben und dann eine geeignete rechte Seite s und Konstanten S_L, S_R berechnen und die Anzahl der Teilintervalle n variieren.

Es wird die Funktion

$$u(z) = e^{\lambda z} \quad (2.32)$$

als Lösung für die lineare DGL (Gleichung (2.4)) mit den Ableitungen

$$\frac{\partial u(z)}{\partial z} = \lambda \cdot e^{\lambda z} \quad (2.33)$$

$$\frac{\partial^2 u(z)}{\partial z^2} = \lambda^2 \cdot e^{\lambda z} \quad (2.34)$$

als Lösung der Differenzialgleichung (Gleichung (2.4)) angesetzt.

Für die Konstanten d, D, k , werden die Werte (Tabelle 3.1) festgelegt.

Die Rekombinationsraten S_L, S_R der Randbedingungen Gleichung (2.3) wird durch

$$S_L = \frac{\partial u}{\partial z} \cdot \frac{D}{u(0)} \quad S_R = -\frac{\partial u}{\partial z} \cdot \frac{D}{u(d)} \quad (2.35)$$

bestimmt.

Die dafür benötigte Ladungsträgerdichte

$$u(0) = e^{\lambda \cdot 0} = 1, \quad u(d) = e^{\lambda \cdot d} \quad (2.36)$$

wird an den Rändern ausgewertet.

Mit Gleichungen (2.33) und (2.36) folgt für Gleichung (2.35)

$$S_L = \lambda \cdot e^{\lambda \cdot 0} \cdot \frac{D}{e^{\lambda \cdot 0}} = \lambda D, \quad S_R = -\lambda \cdot e^{\lambda d} \cdot \frac{D}{e^{\lambda \cdot d}} = -\lambda D \quad (2.37)$$

Um die Numerische Methode zu testen wird die analytische Lösung der Differenzialgleichung mit der Numerischen verglichen. Dazu wird die Anzahl der Teilintervalle n variiert und an den Stützstellen mit der analytischen Lösung verglichen.

Für die lineare, stationäre Differenzialgleichung (Gleichung (2.4)) mit der angesetzten Lösung (Gleichung (2.33)) folgt

$$D \cdot \lambda^2 e^{\lambda z} - k u(z) = -s(z) \quad (2.38)$$

und durch die Fundamentallösung (Gleichung (2.32)) folgt die rechte Seite

$$\begin{aligned} D \cdot \lambda^2 e^{\lambda z} - k e^{\lambda z} &= -s(z) \\ - (D \lambda^2 - k) \cdot e^{\lambda z} &= s(z) \end{aligned} \quad (2.39)$$

Um die Methode zu Testen werden die Parameter

$$S_L = -0.3, \quad S_R = 0.3, \quad \lambda = -1 \quad (2.40)$$

für $\lambda = -1$ bestimmt.

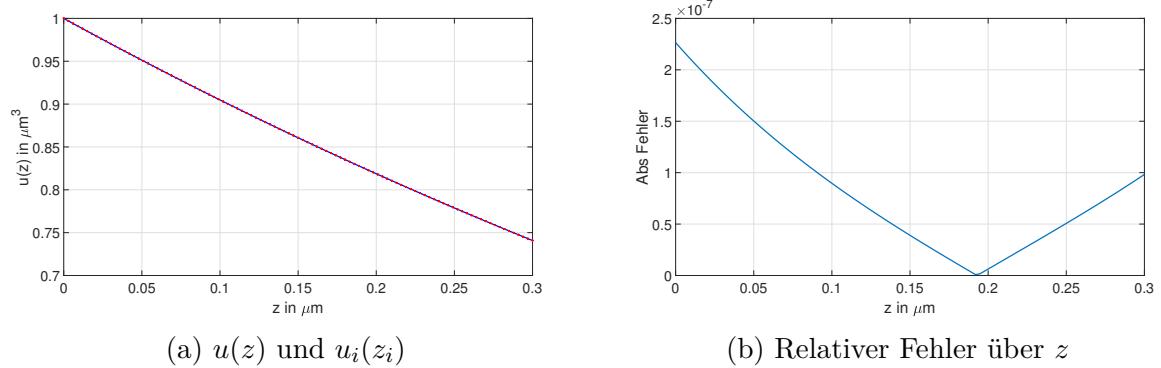


Abbildung 2.2: Vergleich Numerische- mit Analytischelösung der Differentialgleichung bei $N = 100$, $\lambda = -1$

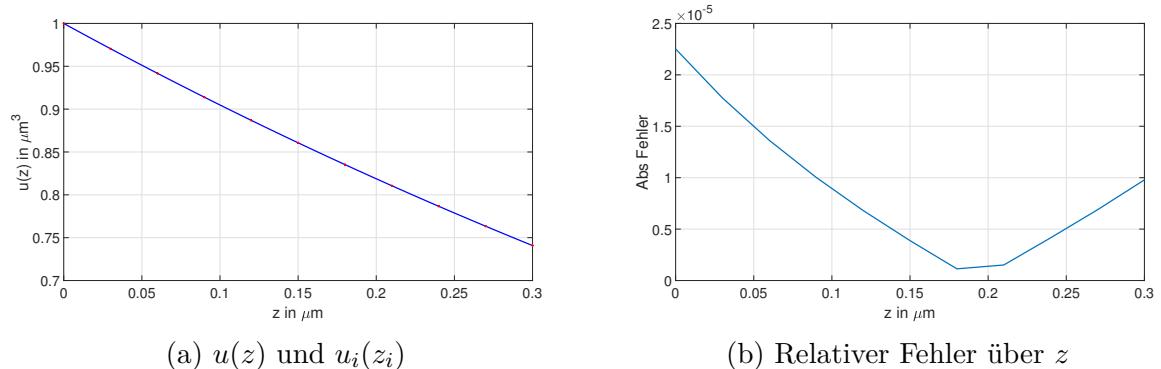


Abbildung 2.3: Vergleich Numerische- mit Analytischelösung der Differentialgleichung bei $N = 10$, $\lambda = -1$

Für ein $\lambda = -100$ wird die Abweichung der Numerischen Simulation deutlicher:

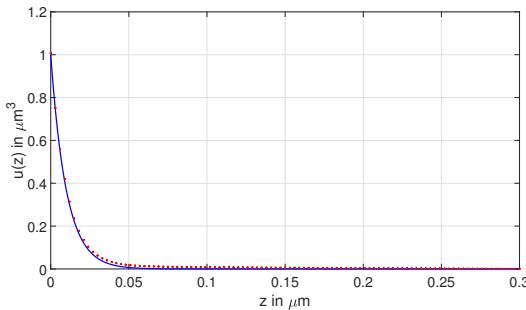
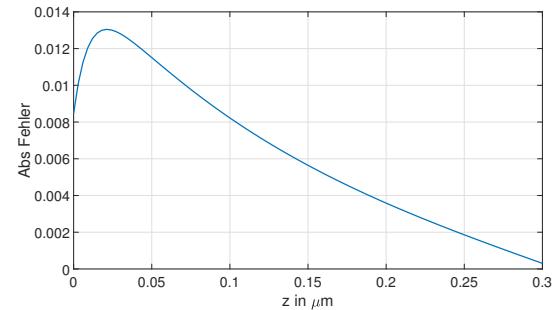
(a) $u(z)$ und $u_i(z_i)$ (b) Relativer Fehler über z

Abbildung 2.4: Vergleich Numerische- mit Analytischelösung der Differentialgleichung bei $N = 100, \lambda = -100$

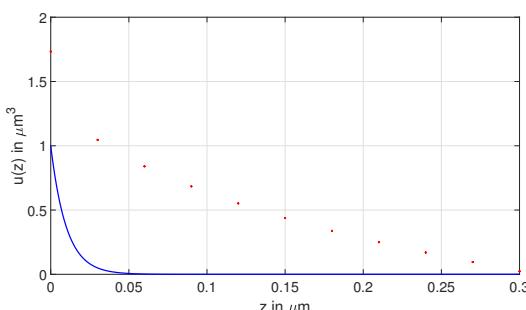
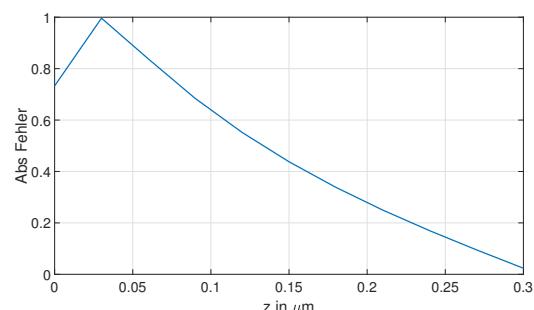
(a) $u(z)$ und $u_i(z_i)$ (b) Relativer Fehler über z

Abbildung 2.5: Vergleich Numerische- mit Analytischelösung der Differentialgleichung bei $N = 10, \lambda = -100$

Codeauszug 2.1: Testfunktion

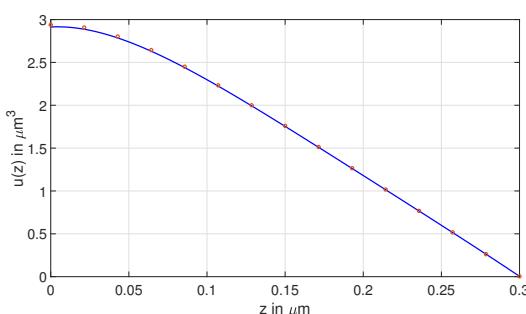
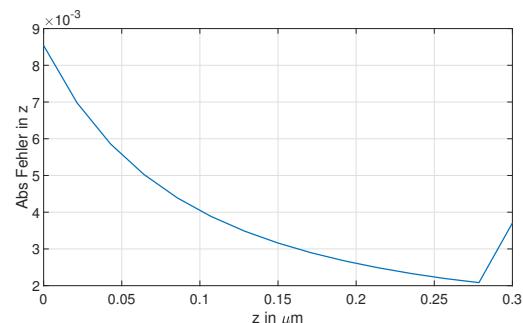
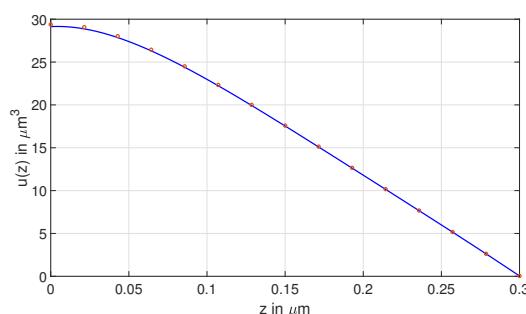
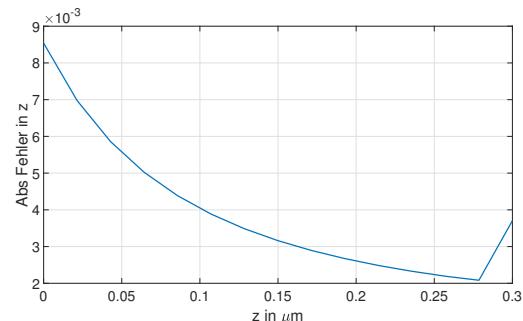
```
1 %%  
2 %Neuer Test  
3 close all;  
4 lamda=-100;  
5 ufunc=@(z) exp(lamda*z);  
6 N=10;  
7 c=konstanten;  
8 ud=exp(lamda*c.d);  
9 u0=exp(lamda*0);  
10  
11 c.SL=lamda*exp(lamda*0)*c.D/u0;  
12 c.SR=-lamda*exp(lamda*c.d)*c.D/ud;  
13  
14 s=@(z) -(c.D*lamda^2-c.k)*exp(lamda*z);  
15  
16  
17 [z,u]=stationaer_lin(s,N,c);  
18 z2=linspace(0,c.d,1000);  
19 ufund=ufunc(z2);  
20  
21 plot(z2,ufund,'b-',z,u,'r*',LineWidth=4,MarkerSize=6)  
22 ylabel("u(z) in \mu^3")  
23 xlabel("z in \mu")  
24 set(gca,'FontSize',45)  
25 grid on  
26 figure  
27  
28 ufund=ufunc(z);  
29 ud=abs(ufund-u);  
30 plot(z,ud,LineWidth=4)  
31 xlabel("z in \mu")  
32 ylabel("Abs Fehler")  
33 set(gca,'FontSize',45)  
34 grid on;
```

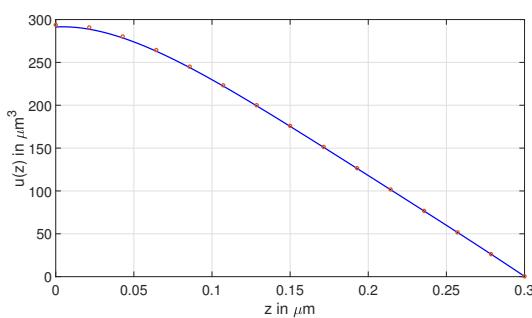
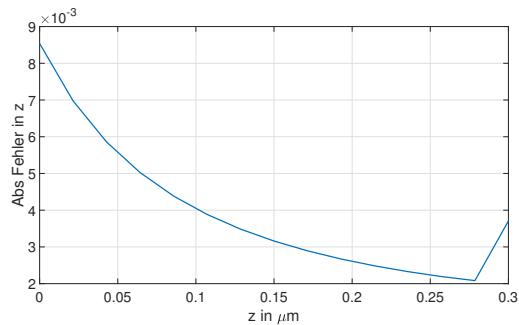
Aufgabe 7. : Berechnen Sie mit Hilfe der obigen Routine die Lösung für die Fälle

$$s(z) = S_0 \cdot e^{-\alpha z}, \quad S_0 = 10^2, 10^2, 10^4 \frac{1}{\mu\text{m}^3 \mu\text{s}} \quad (2.41)$$

Bestimmen Sie näherungsweise experimentell ein geeignetes N , so dass der relative Fehler in u_i maximal 1% beträgt, und stellen Sie die Lösung graphisch dar.

Um ein passendes N für die Bedingung, den relativen Fehler kleiner als 1% zu halten, zu ermitteln, wird über verschiedene N iteriert und der jeweiligen korrespondierenden maximalen Fehler bestimmt.

(a) $u(z)$ und $u_i(z_i)$ (b) Relativer Fehler über z Abbildung 2.6: u_i für $S_0 = 10^2$ und der relative Fehler bei $N = 14$ (a) $u(z)$ und $u_i(z_i)$ (b) Relativer Fehler über z Abbildung 2.7: u_i für $S_1 = 10^3$ und der relative Fehler bei $N = 14$

(a) $u(z)$ und $u_i(z_i)$ (b) Relativer Fehler über z Abbildung 2.8: u_i für $S_1 = 10^4$ und der relative Fehler bei $N = 14$

Codeauszug 2.2: Testfunktion

```

1 close all
2 c=konstanten;
3 S0=[10^2 10^3 10^4]; %1/ $\mu\text{m}^3$ us
4 s=@(z) S0(3)*exp(-c.a*z);
5
6
7 max_rel_F=10^-3;
8 multi=2;
9 for N=10:2:10000
10
11 [z1,u1] = stationaer_lin(s, N);
12 [z2,u2] = stationaer_lin(s, N*multi);
13
14 %|F1-F2|/|(Beste Näherung)|
15 d_u_rel = abs(u1-u2(1:multi:end)) ./ abs(u2(1:multi:end));
16 rel_F = max(d_u_rel); % max Fehler suchen
17 %maxerr(i) = norm((u1-u2s)/u2s,inf);
18 if rel_F < max_rel_F % Bis Fehler kleiner max Fehler
19     break;
20 end
21
22 end

```

```

23 plot(z2,u2,'b-',z1,u1,'o',LineWidth=4,MarkerSize=10)
24 ylabel("u(z) in \mu^3")
25 xlabel("z in \mu")
26 grid on
27 set(gca,'FontSize',45)
28
29 figure
30 plot(z1,d_u_rel,LineWidth=4)
31 grid on
32 set(gca,'FontSize',45)
33 xlabel("z in \mu")
34 ylabel("Abs Fehler in z")
35 figure
36 %plot(2:2:N,rel_F(1:2:N))

```

2.2 Nicht Lineare stationäre Gleichung

Im zweiten Schritt soll nun die volle nichtlineare Gleichung (2.2) mit Randbedingung Gleichung (2.3) gelöst werden. Analog zum letzten Abschnitt wird ein nichtlineares Gleichungssystem aufgestellt, das mithilfe des Newton- Verfahrens aus der Belegarbeit gelöst werden soll. Gehen Sie wie folgt vor.[1]

Aufgabe 1. : Leiten Sie die nichtlinearen Gleichungen für die gesuchten Werte u_0, u_1, \dots, u_N her und stellen Sie sie in Form

$$\mathbf{F}(\mathbf{u}) = \mathbf{b}$$

dar. Berücksichtigen Sie die Randbedingungen wie im letzten Abschnitt. [1]

Nichtlineare DGL:

$$D \frac{\partial^2 u}{\partial z^2} - (k_1 + k_2 N_D)u - k_2 u^2 = -s(z) \quad (2.42)$$

Diskretisierung der DGL:

$$D \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} - k \cdot u_i - k_2 \cdot u_i^2 = z_i \quad (2.43)$$

Mit den Randbedingung:

$$D \cdot \frac{\partial u}{\partial z}(0) = S_L u(0), \quad D \frac{\partial u}{\partial z}(d) = -S_R u(d) \quad (2.44)$$

Und den Approximationen der ersten Ableitung der Randbedingungen:

$$u'(0) \approx \frac{u_1 - u_{-1}}{2h} \quad u'(d) \approx \frac{u_{N+1} - u_{N-1}}{2h} \quad (2.45)$$

Damit folgt für die Randbedingung:

$$D \cdot \frac{u_1 - u_{-1}}{2h} = S_L u_0, \quad D \frac{u_{N+1} - u_{N-1}}{2h} = -S_R u_N \quad (2.46)$$

umgestellt nach u_{-1}

$$u_{-1} = -\frac{S_L 2h}{D} \cdot u_0 + u_1 \quad (2.47)$$

umgestellt nach u_{N+1}

$$u_{N+1} = -\frac{S_R 2h}{D} \cdot u_N + u_{N-1} \quad (2.48)$$

Damit folgt für die Funktion $\mathbf{F}(\mathbf{u}) = \mathbf{b}$:

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

Vereinfacht zu :

$$\begin{aligned}
 F_0 &= 2 \cdot \frac{D}{h^2} u_1 - \frac{S_L 2h + 2D + kh^2}{h^2} u_0 - k_2 u_0^2 \\
 &\quad \vdots \\
 F_i &= \frac{D}{h^2} u_{i+1} - \frac{2D + kh^2}{h^2} u_i + \frac{D}{h^2} \cdot u_{i-1} - k_2 u_i^2 \\
 &\quad \vdots \\
 F_N &= -\frac{2D + kh^2 + S_R 2h}{h^2} u_N + 2 \frac{D}{h^2} u_{N-1} - k_2 u_N^2
 \end{aligned}$$

Aufgabe 2. : Implementieren Sie eine Routine zur Berechnung von \mathbf{F} der letzten Teilaufgabe.[1]

Aufgabe 3. : Berechnen Sie die Jacobi-Matrix $D\mathbf{F}$ von \mathbf{F} bezüglich \mathbf{u} und geben Sie sie an

```

1 function [F] = fd_nonlin(u,N)
2 % Eingabe:
3 % u Vektor u der Größe (N + 1)
4 % N Anzahl von Teilintervallen N.
5 % Ausgabe:
6 % F Vektor F(u) der Größe (N + 1)
7
8 F = zeros([N+1 1]);
9
10 c=konstanten; % Lade Konstanten
11
12 h=c.d/N;% Schrittgröße
13
14 % Matrix besetzen
15 x1=u(1:N-1).*((c.D)/ h ^2 );
16 x2=u(2:N).*((-2*c.D)/h^2-c.k);
17 x3=u(3:N+1).*(c.D/h^2);
18 x4=u(2:N).^2*-(c.k2);
19 F(2:N)=x1+x2+x3+x4;
20
21
22 % Randwerte in Matrix einfügen
23 F(1)=2*c.D/h^2*u(2)-((c.SL*2*h+2*c.D+c.k*h^2)/(h^2))*u(1)-c.k2*u(1)^2;
24 F(N+1)=-(2*c.D+c.k*h^2+c.SR*2*h)/(h^2)*u(N+1)+2*(c.D)/(h^2)*u(N)-c.k2*u(N+1)^2;
25 end

```

$$J_F(u) = \begin{bmatrix} \frac{\partial F_0(u)}{\partial u_0} & \frac{\partial F_0(u)}{\partial u_1} & \cdots & \frac{\partial F_0(u)}{\partial u_N} \\ \frac{\partial F_1(u)}{\partial u_0} & \frac{\partial F_1(u)}{\partial u_1} & \cdots & \frac{\partial F_1(u)}{\partial u_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial F_N(u)}{\partial u_0} & \frac{\partial F_N(u)}{\partial u_1} & \cdots & \frac{\partial F_N(u)}{\partial u_N} \end{bmatrix} \quad (2.49)$$

$$DF(u) = \quad (2.50)$$

$$\begin{bmatrix} -\frac{S_L 2h + 2D + kh^2}{h^2} - 2 \cdot k_2 u_0 & 2 \cdot \frac{D}{h^2} & & \\ \frac{D}{h^2} & -\frac{2D + kh^2}{h^2} - 2 \cdot k_2 u_1 & \frac{D}{h^2} & \\ & \ddots & \ddots & \ddots \\ & \frac{D}{h^2} & -\frac{2D + kh^2}{h^2} - 2 \cdot k_2 u_{N-1} & \frac{D}{h^2} \\ & & 2 \frac{D}{h^2} & -\frac{2D + kh^2 + S_R 2h}{h^2} - 2 \cdot k_2 u_N \end{bmatrix} \quad (2.51)$$

Aufgabe 4. Implementieren Sie eine Routine zur Berechnung der Jacobi-Matrix DF von F der letzten Teilaufgabe. Verwenden Sie dabei dünnbesetzte Matrizen.

```

1 function J = fd_nonlin_jac(u,N)
2 %Eingabe:
3 %
4 % u Vektor u der Größe (N + 1) x 1
5 % N Anzahl von Teilintervallen N.
6 % Ausgabe:
7 % J Jacobi-Matrix DF(u) der Größe (N + 1) x (N + 1) als sparse matrix
8
9 c = konstanten; %Konstanten
10 h=c.d/N; %Schrittweite
11
12 % Matrix besetzen
13 a=c.D/h^2+u*0;
14 b=-(2*c.D+c.k*h^2)/(h^2)+u*0;
15 J=spdiags([a b-2*c.k2*u a], [-1, 0, 1], N+1, N+1);
16
17 % Randbedingungen einfügen
18
19 % Linke Seite
20 J(1,1)=-(c.SL*2*h+2*c.D+c.k*h^2)/h^2-2*c.k2*u(1);
```

```

21 J(1,2)=2*c.D/h^2;
22 %Rechte Seite
23 J(N+1,N+1)=-(c.SR*2*h+2*c.D+c.k*h^2)/h^2-2*c.k2*u(N+1);
24 J(N+1,N)=2*c.D/h^2;
25 end

```

Aufgabe 5. : Implementieren Sie eine Routine zur Lösung des Gleichungssystems (8) unter Verwendung `fd_nonlin` und `fd_nonlin_jac` und des Newton-Verfahrens wie in der Belegaufgabe. Wählen Sie einen geeigneten Startwert.

```

1 function [z,u]=stationaer_nonlin(s,N,tol,nmax)
2 % Eingabe:
3 % s Funktionshandle auf Funktion s(z)
4 % function sz=s(z) mit Spaltenvektoren z und sz
5 % N Anzahl von Teilintervallen N.
6 % tol Toleranz für Newton-Verfahren
7 % nmax maximale Anzahl an Schritten des Newton-Verfahrens
8 % Ausgabe:
9 % z Knotenpunkte (z0,z1,..., zN) der
10
11 c=konstanten; % Konstanten
12 z = linspace(0, c.d, N+1)'; % Knotenpunkte gleichmäßig auf Bereich ←
     verteilen
13 z0=ones(N+1,1); % Startvektor
14
15 function [y, dy] = f(u,N,z)
16     y = fd_nonlin(u,N) +s(z);
17     dy = fd_nonlin_jac(u,N);
18 end
19
20 [u, ~, ~]=newton(@(u)f(u,N,z) , z0, tol, nmax);
21
22 end

```

Aufgabe 6. : Testen Sie Ihre Routine, indem Sie sich eine Funktion $u(z)$ und Konstanten d, D, k vorgeben und dann eine geeignete rechte Seite s und Konstanten S_L, S_R berechnen und die Anzahl der Teilintervalle n variieren.

Vergleichbar mit Abschnitt 2.1 wird eine Lösung für die DGL angesetzt.

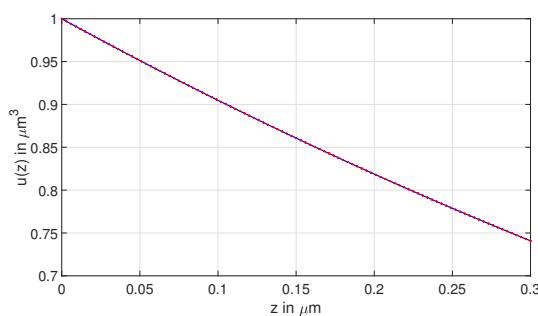
$$u(z) = e^{\lambda z} \quad (2.52)$$

Zusätzlich wird der quadratische Term $k_2 u^2$ der Gleichung (2.38) für die Berechnung der rechten Seite hinzugefügt.

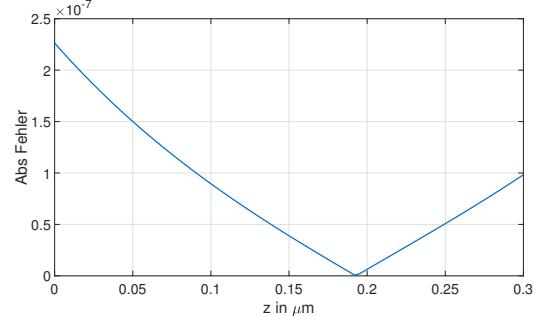
$$D \cdot \lambda^2 e^{\lambda z} - k u(z) - k_2 u^2 = -s(z) \quad (2.53)$$

Mit Gleichung (2.52) folgt für Gleichung (2.54)

$$D \cdot \lambda^2 e^{\lambda z} - k e^{\lambda z} - k_2 e^{2\lambda z} = -s(z) \quad (2.54)$$



(a) $u(z)$ und $u_i(z_i)$



(b) Relativer Fehler über z

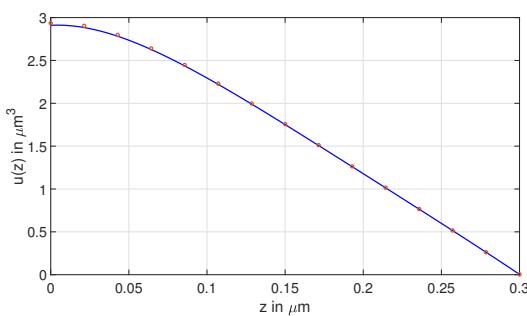
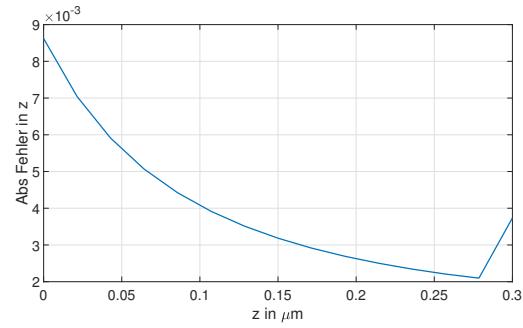
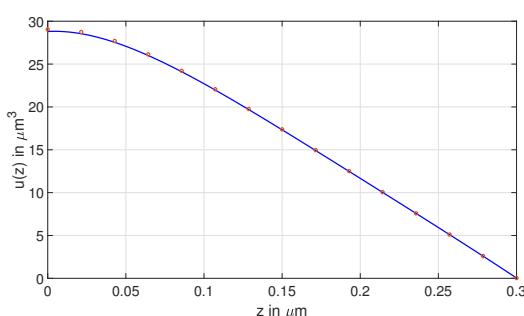
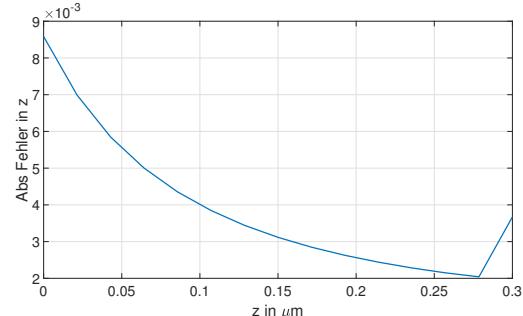
Abbildung 2.9: Vergleich Numerische- mit Analytischelösung der Differentialgleichung bei $N = 100, \lambda = -1$

Aufgabe 7. : Berechnen Sie mit Hilfe der obigen Routine die Lösung für die Fälle

$$s(z) = S_0 \cdot e^{-\alpha z}, \quad S_0 = 10^2, 10^2, 10^4 \frac{1}{\mu\text{m}^3 \mu\text{s}} \quad (2.55)$$

Bestimmen Sie näherungsweise experimentell ein geeignetes N , so dass der relative Fehler in u_i maximal 1% beträgt, und stellen Sie die Lösung graphisch dar. Vergleichen Sie die Lösung mit den entsprechenden Lösungen der linearen Gleichung.

Vergleicht man die Ergebnisse aus Abschnitt 2.1 mit den aus Abschnitt 2.2 wird deutlich das sich die Anzahl der Teilintervalle N für die geforderte Genauigkeit nicht ändert. Ebenso wird deutlich, dass der quadratische Term wenig Auswirkung auf die Signalverläufe hat.

(a) $u(z)$ und $u_i(z_i)$ (b) Relativer Fehler über z Abbildung 2.10: u_i für $S_0 = 10^2$ und der relative Fehler bei $N = 14$ (a) $u(z)$ und $u_i(z_i)$ (b) Relativer Fehler über z Abbildung 2.11: u_i für $S_0 = 10^3$ und der relative Fehler bei $N = 14$

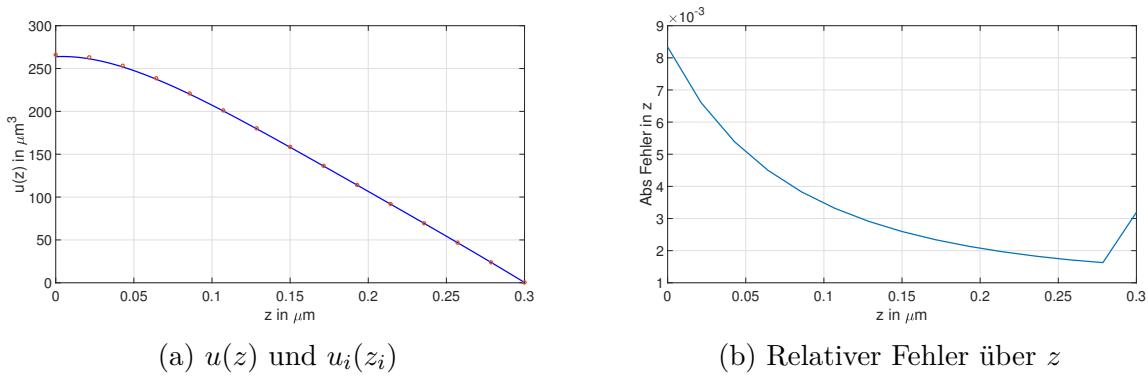
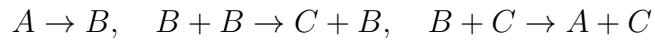
(a) $u(z)$ und $u_i(z_i)$ (b) Relativer Fehler über z

Abbildung 2.12: u_i für $S_0 = 10^4$ und der relative Fehler bei $N = 14$

3 Implizite Einschrittverfahren

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 [3] dienen:



Die Dynamik der Konzentrationen der einzelnen Komponenten können durch Anfangswertproblem

$$A : y'_1 = -0.04 \cdot y_1 + 10^4 \cdot y_2 y_3 \quad (3.1)$$

$$B : y'_2 = 0.04 \cdot y_1 - 10^4 \cdot y_2 y_3 - 3 \cdot 10^7 y_2^2 \quad (3.2)$$

$$C : y'_3 = 3 \cdot 10^7 y_2^2 \quad (3.3)$$

(3.4)

mit den Anfangswerten

$$y_1(0) = 1, y_2(0) = 0, y_3(0) = 0$$

modelliert werden. Anhand dieses Systems sollen im folgenden steife Differenzialgleichungen untersucht und ein effizientes numerisches Verfahren entwickelt werden [1].

3.1 Entwicklung

Ein allgemeines System von Anfangswertproblemen für eine Funktion y kann als folgendes geschrieben werden:

$$y' = f(t, y), \quad t \in [a, b], \quad y(a) = y_a \quad (3.5)$$

Der Wert von y wird näherungsweise an den Zeitpunkten berechnet:

$$y^i \approx y(t_i), \quad i = 0, 1, \dots, N \quad (3.6)$$

Das implizite Euler-Verfahren:

$$y^{(i+1)} = y^{(i)} + h f\left(t_{i+1}, y^{(i+1)}\right) \quad (3.7)$$

Die implizite Trapezregel:

$$y^{(i+1)} = y^{(i)} + \frac{h}{2} \left[f\left(t_i, y^{(i)} + f(t_{i+1}, y^{(i+1)})\right) \right] \quad (3.8)$$

Aufgabe 1. Erarbeiten Sie sich Abschnitt 8.4 aus [2] und fassen Sie in eigenen Worten zusammen, was Sie unter steifen Differenzialgleichung verstehen.

Unter einer steifen Differenzialgleichung versteht man eine Gleichung mit bestimmten numerischen Methoden, die besonders große Störungsfaktoren am Ausgang aufweisen, sofern die Schrittweite nicht allzu klein ausgewählt wird.

Aufgabe 2: Setzen Sie in den Gleichungen des impliziten Euler-Verfahrens und der impliziten Trapezregel

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

und formulieren Sie jeweils die Gleichung für als Nullstellenproblem:

$$\mathbf{F}_{\text{euler}}(\mathbf{z}) = 0, \quad \mathbf{F}_{\text{trapez}}(\mathbf{z}) = 0$$

$$y^{(i+1)} = y^{(i)} + h f(t_{(i+1)}, y^{(i+1)}) \quad (3.9)$$

und die implizite Trapezregel:

$$y^{(i+1)} = y^{(i)} + \frac{h}{2} [f(t_i, y^{(i)}) + f(t_{(i+1)}, y^{(i+1)})] \quad (3.10)$$

wir setzen nun

$$y^{(i+1)} = y^{(i)} + z \quad (3.11)$$

in:

$$y^{(i)} + z = y^{(i)} + h f(t_i + 1, y^{(i)} + z) \quad (3.12)$$

und in:

$$y^{(i)} + z = y^{(i)} + \frac{h}{2} [f(t_i, y^{(i)}) + f(t_{i+1}, y^{(i)} + z)] \quad (3.13)$$

ein.

Formulieren wir nun beides als Nullstellenproblem für erhaltenen wir

$$F_{\text{euler}}(z) = z - h f(t_{i+1}, y^{(i)} + z) = 0 \quad (3.14)$$

und

$$F_{\text{trapez}}(z) = z - \frac{h}{2} [f(t_i, y^{(i)}) + f(t_{i+1}, y^{(i)} + z)] = 0 \quad (3.15)$$

Aufgabe 3. : Bestimmen Sie die Jacobi-Matrizen $D\mathbf{F}_{\text{euler}}$ und $D\mathbf{F}_{\text{trapez}}$ von $\mathbf{F}_{\text{euler}}$ und $\mathbf{F}_{\text{trapez}}$ in Abhängigkeit von $D_y f$, d.h. der Jacobi-Matrix von f bezüglich y .

$$D \cdot F_{\text{euler}}(z) = D \cdot z - h \cdot D_y \cdot f(t_{i+1}, y^{(i)} + z) \quad (3.16)$$

für das implizite Euler-Verfahren und

$$D \cdot F_{\text{trapez}}(z) = \frac{h}{2} \cdot [D_y \cdot f(t_{i+1}, y^{(i)}) + h \cdot D_y f(t_{i+1}, y^{(i)} + z)] - D \cdot z \quad (3.17)$$

für das implizite Trapez-Verfahren,

wobei

$$D_f = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \dots & \frac{\partial f_1}{\partial y_n} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \dots & \frac{\partial f_2}{\partial y_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_n}{\partial y_1} & \frac{\partial f_n}{\partial y_2} & \dots & \frac{\partial f_n}{\partial y_n} \end{bmatrix} \quad (3.18)$$

und

$$Dz = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 1 \end{bmatrix} \quad (3.19)$$

Aufgabe 4. : Implementieren Sie die folgenden zwei Routinen. Verwenden Sie für die Jacobi-Matrix J dünnbesetzte Matrizen.

```

1 function [F,J] = F_euler(z, ti, h, yi, f, df)
2 % Eingabe:
3 % z Vektor z der Größe k x 1
4 % ti Zeitpunkt ti
5 % h Schrittweite h
6 % yi Vektor y(i) der Größe k x 1
7 % f Funktionshandle für f (t; y)
8 % df Funktionshandle für Dyf(t; y)
9
10 % Ausgabe:
11 % F Vektor Feuler(z) bzw. Ftrapez(z) der Größe k x 1
12 % J Jacobi-Matrix DFeuler(z) bzw. DFtrapez(z) der Größe k x k als ←
13 % sparse matrix.
14 F=z-h*f(ti+h,yi+z);
15
16 J=speye(length(z))-h*df(ti+h,yi+z);
17 end

```

```
1 function[F,J] = F_trapez(z, ti, h, yi, f, df)
2 %Eingabe:
3 %z Vektor z der Größe k x 1
4 %ti Zeitpunkt ti
5 %h Schrittweite h
6 %yi Vektor y
7 %(i) der Größe k x 1
8 %f Funktionshandle für f(t, y)
9 %df Funktionshandle für Dyf(t, y)
10 %Ausgabe:
11 %F Vektor Feuler(z) bzw. Ftrapez(z) der Größe k x 1
12 %J Jacobi-Matrix DFeuler(z) bzw. DFtrapez(z) der Größe k x k als ←
    sparse matrix.
13
14 F = z-0.5*h*(f(ti,yi)+ f(ti+h,yi+z));
15 J=speye(length(z))-0.5*h*df(ti+h,yi+z);
16 end
```

Aufgabe 5. : Implementieren Sie die Löser aufbauend auf den Routinen des letzten Abschnitts und des Newton-Verfahrens aus der Belegarbeit. Als Startwert der Newtoniteration wählen Sie für z den Nullvektor.

```
1 function [t,y]=impl_euler(f,tspan,ya,n,df,tol,nmax)
2 % Eingabe:
3 % f Funktionshandle für f (t; y)
4 % tspan Intervall [a; b]
5 % ya Anfangswert ya
6 % n Anzahl von Teilintervallen.
7 % df Funktionshandle für Dyf (t; y) bezüglich y
8 % tol Toleranz für Newton-Verfahren
9 % nmax maximale Anzahl an Schritten des Newton-Verfahrens
10 % Ausgabe:
11 % t Vektor der Stützpunkte (t0; t1; : : : ; tn) der Größe (n + 1) x 1
12 % y Matrix der approximierten Lösungswerte y(i) mit i = 0; : : : ; ↵
13 % n der Größe (n + 1) x k
13 k=length(ya);
14 h=(tspan(2)-tspan(1))/n;
15 t=zeros(n+1,1);
16 y=zeros(n+1,k);
17 y(1,:)=ya;
18 t(1)=tspan(1);
19 x0=zeros(k,1);
20 for i=1:n
21     t(i+1)=tspan(1)+h*i;
22     % euler= @(z)F_euler(z,t(i),h,y(i),f,df);
23     % z=newton(@(z)F_euler(z,t(i),h,y(i,:)',f,df),x0,tol,nmax);
24     y(i+1,:)=y(i,:)+z';
25 end
```

```
1 function[t,y]=impl_trapez(f,tspan,ya,n,df,tol,nmax)
2 %Eingabe:
3 %f Funktionshandle für f(t, y)
4 %tspan Intervall [a, b]
5 %ya Anfangswert ya
6 %n Anzahl von Teilintervallen.
7 %df Funktionshandle für Dyf(t, y) bezüglich y
8 %tol Toleranz für Newton-Verfahren
9 %nmax maximale Anzahl an Schritten des Newton-Verfahrens
10 %Ausgabe:
11 %t Vektor der Stützpunkte (t0, t1, . . . , tn) der Größe (n + 1) x 1
12 %y Matrix der approximierten Lösungswerte y
13 %(i) mit i = 0, . . . , n der Größe (n + 1) x k
14 k=length(ya);
15 h=(tspan(2)-tspan(1))/n;
16 t=zeros(n+1,1);
17 y=zeros(n+1,k);
18 y(1,:)=ya';
19 t(1)=tspan(1);
20 x0=zeros(k,1);
21 for i=1:n
22     t(i+1)=tspan(1)+h*i;
23     [z,~,~] = newton(@(z) F_trapez(z, ←
24         t(i),h,y(i,:)',f,df),x0,tol,nmax);
25     y(i+1,:)=y(i,:)+z';
26 end
```

Aufgabe 6. : Bestimmen Sie wie in den Übungsblättern anhand des Anfangswertproblems $y' = -y$ mit Anfangsbedingung $y(0) = 1$ auf dem Intervall $[0, 1]$ graphisch die Ordnung der beiden Verfahren.

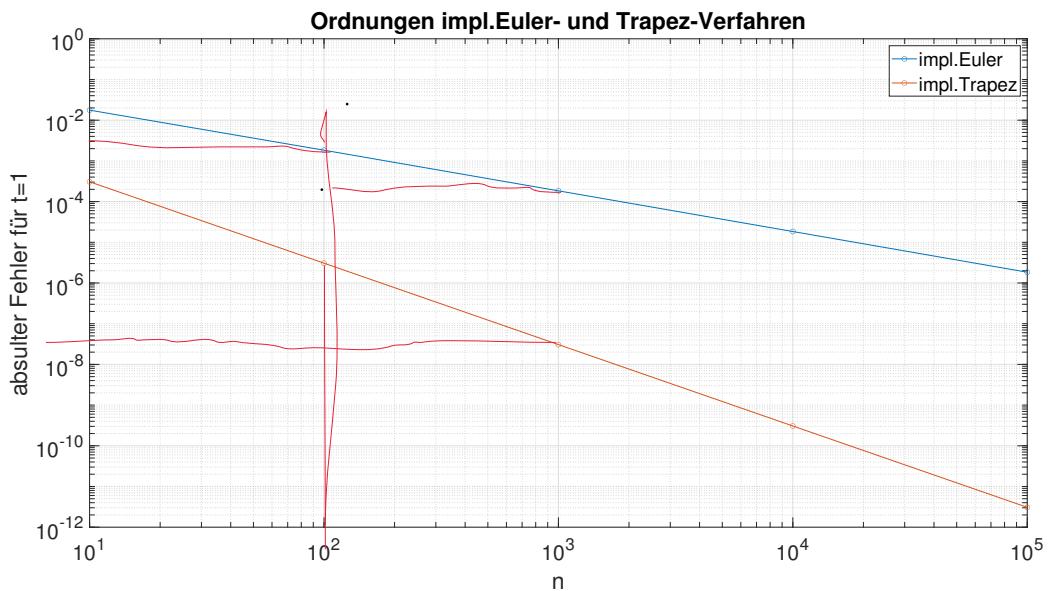


Abbildung 3.1: die Ordnungen von impliziten Euler- und Trapez-Verfahren

Aufgabe 7. : Testen Sie Ihre Routine anhand eines geeigneten nichtlinearen, zeit-abhängigen Systems von Anfangswertproblemen.

$$\begin{aligned} y' &= \begin{bmatrix} -2t \cdot y_1 - 3y_2 \\ 2y_1 + t^2 \cdot y_2 \end{bmatrix} \quad \text{mit } y(0) = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, t = [0, 2], \end{aligned} \quad (3.20)$$

Als Test für das System wird folgendes Gleichungssystem(nichtlinear) entwickelt:

Codeauszug 3.1: Test zum Lösen von nichtlinearen, zeitabhängigen System von ANfangswertproblemen

```

1 close all
2 N = 1e+3;
3 y_a =[2;1];
4 t_span = [0,2];
5 tol = 10e-6;
6 N_max=100;
7
8 fun = @(t,y) [-3*t*y(1)-2*y(2);y(1)+t.^2*y(2)];
9
10 syms y1 y2 t
11
12 % syms: Erzeugt symbolische skalare Variablen var1 ... varN vom Typ ←
13 % sym.
14 % Verschiedene Variablen sind durch Leerzeichen zu trennen.
15 j_fun = @(t,y) [-3*t,-2;1,2*t.^2];
16
17 [t_fun_euler,y_fun_euler] = ←
18     impl_euler(fun,t_span,y_a,N,j_fun,tol,N_max);
19 [t_fun_trapez,y_fun_trapez] = ←
20     impl_trapez(fun,t_span,y_a,N,j_fun,tol,N_max);
21 figure

```

```
21 plot(t_fun_euler,y_fun_euler);
22 title("Die Gleichung mit dem impl. Euler-Verfahren")
23 xlabel("t");
24 ylabel("y");
25 set(gca,'FontSize',25)
26 grid on
27
28 figure
29 plot(t_fun_trapez,y_fun_trapez);
30 title("Die Gleichung mit dem impl. Trapez-Verfahren")
31 xlabel("t");
32 ylabel("y");
33 set(gca,'FontSize',25)
34 grid on
```

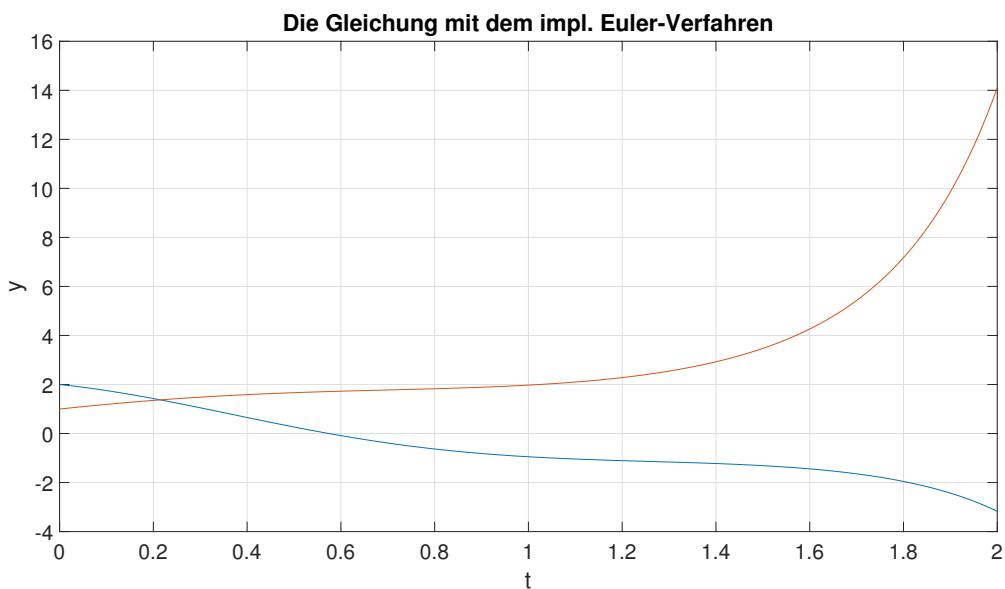


Abbildung 3.2: das implizit Euler-Verfahren

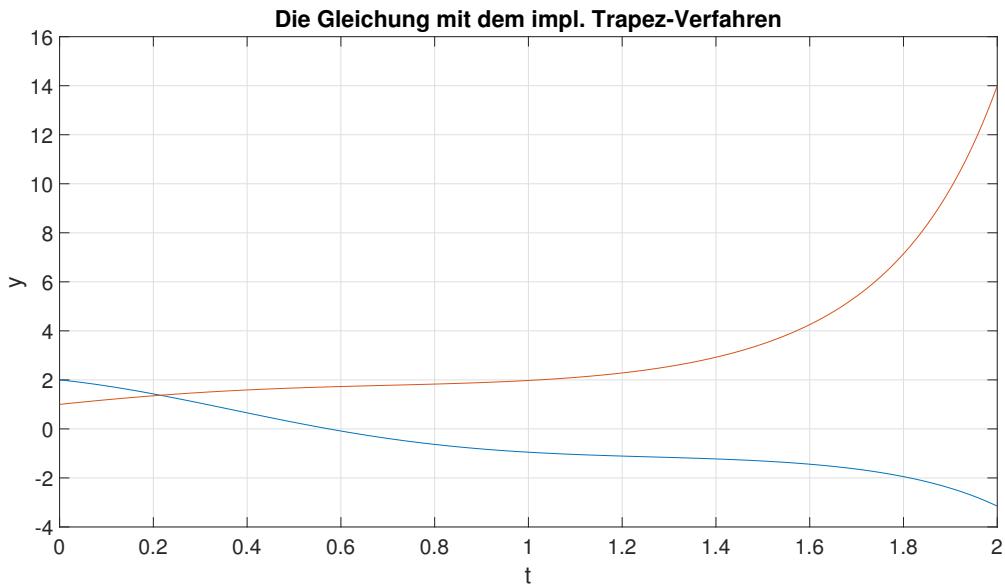


Abbildung 3.3: das implizit Trapez-Verfahren

3.2 Anwendung

Aufgabe 1. : Formulieren Sie Gleichung (9) als System der Form (11) mit Funktion $f_{\text{chem}}(t; y)$

$$y' = f(t, y) = \begin{bmatrix} -0.04y_1 + 10^4 y_2 y_3 \\ 0.04y_1 - 10^4 y_2 y_3 - 3 \cdot 10^7 y_2^2 \\ 3 \cdot 10^7 y_2^2 \end{bmatrix} \quad (3.21)$$

mit $t \in [0, \sim], y(0) =$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (3.22)$$

Aufgabe 2. : Implementieren Sie die Funktion als Routine $d_y = f_{\text{chem}}(t, y)$

```
1 % dy=f_chem(t,y)
2 %-----
```

```

3 % Eingabe:
4 % t Zeit t
5 % y Vektor y der Größe 3 x 1
6 %-----
7 % Ausgabe:
8 % dy Vektor fchem(t; y) der Größe 3 x 1
9
10 function dy = f_chem(t, y)
11 a = 0.04;
12 b = 10^4;
13 c = 10^7;
14 dy(1,1) = -a*y(1) + b*y(2)*y(3);
15 dy(2,1) = a*y(1) - b*y(2)*y(3) - 3*c*y(2)^2;
16 dy(3,1) = 3*c*y(2)^2;
17 end

```

Aufgabe 3. : Berechnen Sie die Jacobi-Matrix $D_y f_{\text{chem}}(t, y)$ von f_{chem} bezüglich y

$$D_y f_{\text{chem}}(t, y) = \begin{bmatrix} -0.04 & 10^4 y_3 & 10^4 y_2 \\ 0.04 & -10^4 y_3 - 6 \cdot 10^7 y_2 & -10^4 y_2 \\ 0 & 3 \cdot 10^7 & 0 \end{bmatrix} \quad (3.23)$$

Aufgabe 4. : Implementieren Sie die Funktion als Routine $J = f_chem_jac(t, y)$

```

1 % J=f_chem_jac(t,y)
2 %-----
3 % Eingabe:
4 % t Zeit t
5 % y Vektor y der Größe 3 x 1
6 %-----
7 % Ausgabe:
8 % J Matrix Dyfchem(t; y) der Größe 3 x 3

```

Tabelle 3.1: Vergleich der Methoden für eine Genauigkeit von 10^{-4}

Methode	N	Abs. Fehler y1 in	Abs. Fehler y2	Abs. Fehler y1
Implizites Euler	26	$0.9051 \cdot 10^{-4}$	$0.0001 \cdot 10^{-4}$	$0.9053 \cdot 10^{-4}$
Implizites Trapez	32	$0.7972 \cdot 10^{-4}$	$0.0263 \cdot 10^{-4}$	$0.7709 \cdot 10^{-4}$
Explizit Euler	1074	$0.8490 \cdot 10^{-4}$	$0.0561 \cdot 10^{-4}$	$0.7928 \cdot 10^{-4}$
Mittelpunkt	640	$0.6314 \cdot 10^{-4}$	$0.1866 \cdot 10^{-4}$	$0.8180 \cdot 10^{-4}$

```

9
10 function J = f_chem_jac(t, y)
11 a = 0.04;
12 b = 10^4;
13 c = 10^7;
14
15 J(1,1) = -a;
16 J(1,2) = b*y(3);
17 J(1,3) = b*y(2);
18
19 J(2,1) = a;
20 J(2,2) = -b*y(3) - 6*c*y(2);
21 J(2,3) = -b*y(2);
22
23 J(3,1) = 0;
24 J(3,2) = 6*c*y(2);
25 J(3,3) = 0;
26 end

```

Aufgabe 5. : Lösen Sie das Anfangswertproblem auf dem Intervall $[0; 1]$ mit den folgenden Integratoren: explizites Euler-Verfahren, Mittelpunktregel (siehe Übungen), implizites Euler-Verfahren und implizite Trapezregel. Bestimmen Sie experimentell jeweils näherungsweise ein geeignetes n , so dass der absolute Fehler jeder einzelnen Komponente und Zeitschritts höchstens 10^{-4} , bzw. 10^{-6} ist. Erstellen Sie eine Tabelle der notwendigen Intervalleinteilungen.

Codeauszug 3.2: Berechnung des Absoluten Fehlers für verschiedenen Methoden

```
1 close all
2 y_a =[1;0;0];
3 t_span = [0,1];
4 N_max=100;
5 tol=10^-8;
6 nmax=100;
7 multi=10;
8 max_rel_F=10^-6;
9 %Implizites Euler
10 for N=2:2:10000
11     [t1,y1] = impl_euler(@f_chem ,t_span,y_a,N,@f_chem_jac,tol,N_max);
12     [t2,y2] = impl_euler(@f_chem,t_span,y_a,N*2,@f_chem_jac,tol,N_max);
13     %|F1-F2|/|(Beste Naehlerung)|
14     d_u_abs = abs(y1(:, :) - y2(1:2:end,:));
15     [u_abs] = max(d_u_abs); % max Fehler suchen
16     if u_abs < max_rel_F %Bis Fehler kleiner max Fehler
17         disp("Implizites Euler")
18         N
19         d_u_abs(N,:)
20         break;
21     end
22 end
23 %Implizites trapez
24 for N=2:2:10000
25     [t1,y1] = impl_trapez(@f_chem ,t_span,y_a,N,@f_chem_jac,tol,N_max);
26     [t2,y2] = ←
27         impl_trapez(@f_chem,t_span,y_a,N*2,@f_chem_jac,tol,N_max);
28     %|F1-F2|/|(Beste Naehlerung)|
29     d_u_abs = abs(y1(:, :) - y2(1:2:end,:));
30     [u_abs] = max(d_u_abs); % max Fehler suchen
31     if u_abs < max_rel_F %Bis Fehler kleiner max Fehler
32         disp("Implizites trapez")
```

```
32      N
33      d_u_abs(N,:)
34      break;
35  end
36 end
37 %Explizites Euler
38 for N=2:2:10000
39 [t1,y1] = loeseE_sys(@f_chem,t_span,y_a,N);
40 [t2,y2] = loeseE_sys(@f_chem,t_span,y_a,N*2);
41 %|F1-F2|/(Beste Naehlerung)|
42 d_u_abs = abs(y1(:, :)-y2(1:2:end,:));
43 [u_abs] = max(d_u_abs); % max Fehler suchen
44 if u_abs < max_rel_F %Bis Fehler kleiner max Fehler
45     disp("Explizit Euler")
46     N
47     d_u_abs(N,:)
48     break;
49 end
50 end
51 %Mittelpunkt
52 for N=2:2:10000
53 [t1,y1] = loeseM_sys(@f_chem,t_span,y_a,N);
54 [t2,y2] = loeseM_sys(@f_chem,t_span,y_a,N*2);
55 %|F1-F2|/(Beste Naehlerung)|
56 d_u_abs = abs(y1(:, :)-y2(1:2:end,:));
57 [u_abs] = max(d_u_abs); % max Fehler suchen
58 if u_abs < max_rel_F %Bis Fehler kleiner max Fehler
59     disp("Mittelpunkt")
60     N
61     d_u_abs(N,:)
62     break;
63 end
64 end
```

Aufgabe 6. : Diskutieren Sie Ihre Ergebnisse im Hinblick auf die Performanz.

Aus 3.2.5 stellt man fest, dass mit kleineren n bei dem Mittelpunktregel-Verfahren auf den Fehler 10^{-4} im Vergleich zum explizit Euler-Verfahren kommt und bei dem impliziten Euler braucht man kleinere Schritte als bei dem implizite Trapez-Verfahren. die Performanz ergibt sich dann als folgendes:

4 Zeitaufgelöste Simulation

Aufgabe 1. : Formulieren Sie das System in der Form $u' = F(t, u)$

Die Randbedingungen

$$D \cdot \frac{\partial u}{\partial z}(0) = S_L u(0), \quad D \frac{\partial u}{\partial z}(d) = -S_R u(d) \quad (4.1)$$

und:

$$k = k_1 + k_2 \cdot N_D$$

$$s_i = s(z_i)$$

$$h = \frac{d}{N}$$

$$u(i) \approx u_i$$

Analog wie im zweiten Abschnitt

$$u' = \frac{u_{i+1} - u_{i-1}}{2h} \quad (4.2)$$

$$u'' = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} \quad (4.3)$$

Anschließend werden die Ableitungen jeweils in die Gleichung sowie Randbedingungen eingesetzt:

$$u'_i(t) = D \frac{u_{i+1}(t) - 2u_i(t) + u_{i-1}(t)}{h^2} - (k_1 + k_2 N_D) u_i(t) - k_2 u_i^2(t) + s_i(t, z_i) \quad (4.4)$$

Dann setzt man die Randbedingungen in die Gleichung für $i = 0$ und $i = N$ ein:

$$u'(t, z_i) = D \frac{u_{i+1}(t) - 2u_i(t) + u_{i-1}(t)}{h^2} - ku_i(t) - k_2 u_i^2(t) + s_i(t, z) \quad (4.5)$$

für $i = 0$ gilt:

$$u'(t, z_i) = D \frac{u_1(t) - 2u_0(t) + u_{-1}(t)}{h^2} - ku_0(t) - k_2 u_0^2(t) + s_0(t, z) \quad (4.6)$$

$$u'(t, z_i) = D \frac{u_1 - 2u_0 + u_1 - \frac{S_L \cdot 2h}{D} u_0}{h^2} - ku_0 - k_2 u_0^2 + s_0(z) \quad (4.7)$$

$$u'(t, z_i) = D \frac{2u_1 - 2u_0 - \frac{S_L \cdot 2h}{D} u_0}{h^2} - ku_0 - k_2 u_0^2 + s_0(z) \quad (4.8)$$

$$u'(t, z_i) = \frac{2Du_1}{h^2} - \frac{2Du_0}{h^2} - \frac{D \cdot S_L 2hu_0}{D \cdot h^2} - ku_0 - k_2 u_0^2 + s_0(z) \quad (4.9)$$

$$u'(t, z_i) = \frac{2Du_1}{h^2} - \frac{2Du_0}{h^2} - \frac{S_L 2u_0}{h} - ku_0 - k_2 u_0^2 + s_0(z) \quad (4.10)$$

$$u'(t, z_i) = u_0(t) \left(-\frac{2D}{h^2} - \frac{2S_L}{h} - k - k_2 u_0(t) \right) + u_1(t) \left(\frac{2D}{h^2} \right) + s_0(t, z_0) \quad (4.11)$$

für $i = N$ gilt:

$$u'(t, z_i) = D \frac{u_{N+1}(t) - 2u_N(t) + u_{N-1}(t)}{h^2} - ku_N(t) - k_2 u_N^2(t) + s_N(z_N) \quad (4.12)$$

$$u'(t, z_i) = D \frac{u_{N-1} - \frac{S_R \cdot 2h}{D} \cdot u_N - 2u_N(t) + u_{N-1}(t)}{h^2} - ku_N(t) - k_2 u_N^2(t) + s_N(z_N) \quad (4.13)$$

$$u'(t, z_i) = u_N(t) \left(-\frac{2S_R}{h} - \frac{2D}{h^2} - k - k_2 u_N(t) \right) + u_{N-1}(t) \left(\frac{2D}{h^2} \right) + s_0(t, z_N) \quad (4.14)$$

Somit kann unsere Gleichung aufgestellt werden:

$$F(t, u) = \begin{bmatrix} u_0(t) \left(-\frac{2D}{h^2} - \frac{2S_L}{h} - k - k_2 u_0(t) \right) + u_1(t) \left(\frac{2D}{h^2} \right) + s_0(t, z_0) \\ D \frac{u_2(t) - 2u_1(t) + u_1(t)}{h^2} - ku_1(t) - k_2 u_1^2(t) + s_1(t, z) \\ \vdots \\ u_N(t) \left(-\frac{2S_R}{h} - \frac{2D}{h^2} - k - k_2 u_N(t) \right) + u_{N-1}(t) \left(\frac{2D}{h^2} \right) + s_0(t, z_N) \end{bmatrix} \quad (4.15)$$

Aufgabe 2. : Stellen Sie Routinen zur Berechnung von $F(t, u)$ und $D_u F(t, u)$ bereit.

Codeauszug 4.1: Berechnung $F(t, u)$

```

1 function f= F_tu(s,n_Ort,z)
2 %F_TU Summary of this function goes here
3 % Detailed explanation goes here
4 f = @(t,u) fd_nonlin(u,n_Ort) + s(t,z);
5 end

```

Codeauszug 4.2: Berechnung $D_u F(t, u)$

```

1 function df = dF_tu(n_Ort)
2 %DF_TU Summary of this function goes here
3 % Detailed explanation goes here
4 df = @(t,u) fd_nonlin_jac(u,n_Ort);
5 end

```

Aufgabe 4. : Verwenden Sie die implizite Trapezregel und simulieren Sie das System für $t \in [-0.05, 0.2]\mu s$

$$s(t, z) = S_0 \cdot e^{-\frac{t^2}{2 \cdot 0.01^2}} \cdot e^{-\alpha z} \quad (4.16)$$

mit

$$S_0 = 10^4, 10^5, 10^6 \frac{1}{\mu m^3 \mu s} \quad (4.17)$$

Stellen Sie die Ergebnisse geeignet dar (siehe mesh).

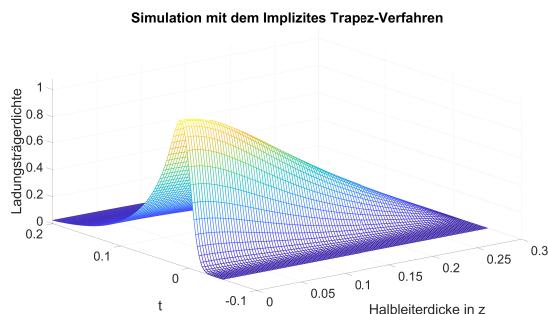
Um die Implementierungen Funktionen zu prüfen wird das Ergebnis des Trapezverfahrens (Abb. 4.1a) mit dem Matlab internen Löser ODE23 verglichen (Abb. 4.1b). Anschließend wird für die verschiedenen S_0 simuliert.

Codeauszug 4.3: Auswertung

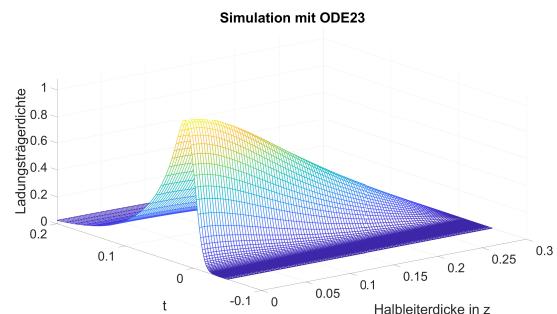
```

1 close all;
2 tspan=[-0.05,0.2];
3 tol=1e-12;
4 nmax=100;
5 n_Ort=100;
6 n_Zeit=100;
7 c=konstanten;
8
9 S0=[10^2 10^3 10^4]; %1/um^3us
10
11 s = @(t,z) S0(3)*exp((-t^2)/(2*0.01^2))*exp(-c.a*z);
12 z=linspace(0,c.d,n_Ort+1)';
13 f=F_tu(s,n_Ort,z);
14 df=dF_tu(n_Ort);
15
16 ya = zeros(n_Ort+1,1);
17 [t,y] = impl_trapez(f,tspan,ya,n_Zeit,df,tol,nmax);
18 mesh(z,t,y,LineWidth=2);
19
20 title('Simulation mit dem Implizites Trapez-Verfahren')
21 xlabel('Halbleiterdicke in z')
```

```
22 ylabel('t')
23 zlabel('Ladungsträgerdichte')
24 set(gca, 'FontSize',40)
25 [t,y]=ode23s(f,tspan,ya);
26 figure
27 mesh(z,t,y,LineWidth=2);
28 title('Simulation mit ODE23')
29 xlabel('Halbleiterdicke in z')
30 ylabel('t')
31 zlabel('Ladungsträgerdichte')
```

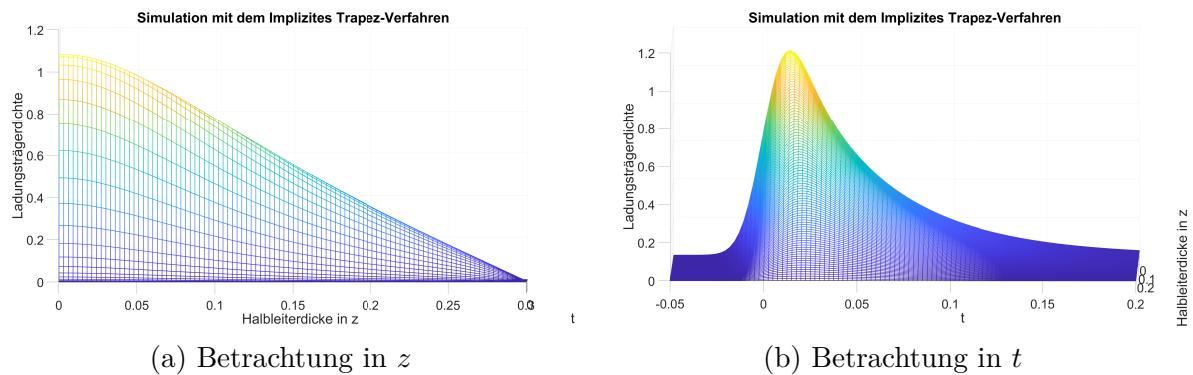
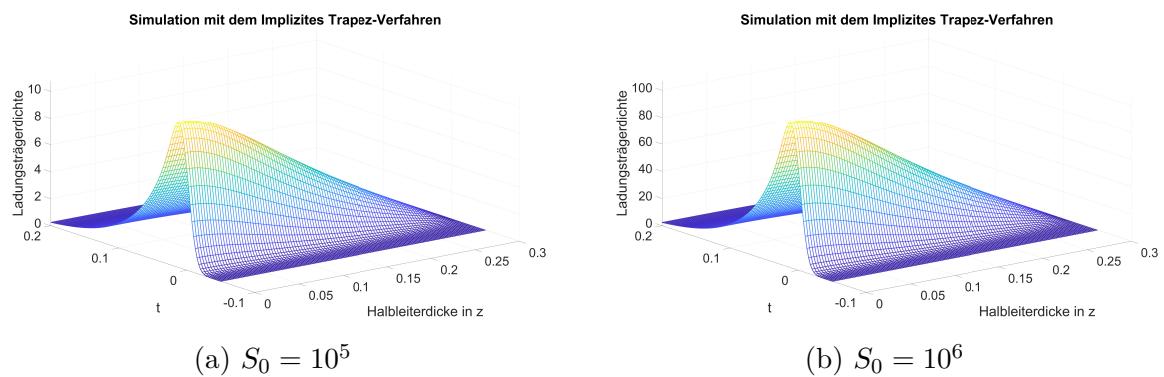


(a) Trapezverfahren



(b) ODE23

Abbildung 4.1: Auswertung der Zeitaufgelösten Simulation mit $S_0 = 10^4$

Abbildung 4.2: Vergleich von z zu t Abbildung 4.3: Auswertung der Zeitaufgelösten Simulation mit $S_0 = 10^5$ und $S_0 = 10^6$

Abbildungsverzeichnis

2.1	Beispieldarstellung: Knotenpunkte mit Schrittweite h	4
2.2	Vergleich Numerische- mit Analytischelösung der Differentialgleichung bei $N = 100, \lambda = -1$	12
2.3	Vergleich Numerische- mit Analytischelösung der Differentialgleichung bei $N = 10, \lambda = -1$	12
2.4	Vergleich Numerische- mit Analytischelösung der Differentialgleichung bei $N = 100, \lambda = -100$	13
2.5	Vergleich Numerische- mit Analytischelösung der Differentialgleichung bei $N = 10, \lambda = -100$	13
2.6	u_i für $S_0 = 10^2$ und der relative Fehler bei $N = 14$	15
2.7	u_i für $S_1 = 10^3$ und der relative Fehler bei $N = 14$	15
2.8	u_i für $S_1 = 10^4$ und der relative Fehler bei $N = 14$	16
2.9	Vergleich Numerische- mit Analytischelösung der Differentialgleichung bei $N = 100, \lambda = -1$	23
2.10	u_i für $S_0 = 10^2$ und der relative Fehler bei $N = 14$	24
2.11	u_i für $S_0 = 10^3$ und der relative Fehler bei $N = 14$	24
2.12	u_i für $S_0 = 10^4$ und der relative Fehler bei $N = 14$	25
3.1	die Ordnungen von impliziten Euler- und Trapez-Verfahren	33
3.2	die Ordnungen von impliziten Euler- und Trapez-Verfahren	35
3.3	die Ordnungen von impliziten Euler- und Trapez-Verfahren	36
4.1	Auswertung der Zeitaufgelösten Simulation mit $S_0 = 10^4$	46
4.2	Vergleich von z zu t	47
4.3	Auswertung der Zeitaufgelösten Simulation mit $S_0 = 10^5$ und $S_0 = 10^6$	47

Literatur

- [1] Prof. Dr. Andreas Zeiser, *Angewandte Mathematik: Projekt Zeitaufgelöste Photolumineszenz*, Moodle.
- [2] K. E. Atkinson und W. Han, *Elementary numerical analysis*, 3rd ed. / Kendall Atkinson, Weimin Han. Chichester: Wiley, 2004, ISBN: 9780471433378. Adresse: <http://worldcatlibraries.org/wcpa/oclc/52418321>.