Aufgaben zur Vorlesung
Numerik II
Wintersemester 2012/13
Übungsblatt 9

W.–J. Beyn
D. Otten

Abgabe: Mittwoch, 12.12.2012, vor Beginn der Übung

Übung: Mi. 12:15–13:45, V5-148

Aufgabe 25: [Lineares Zweischrittverfahren]

a) Bestimmen Sie alle 6 Koeffizienten des linearen Zweischrittverfahrens

$$\frac{1}{h}\sum_{\nu=0}^{2} a_\nu u^{j+\nu} = \sum_{\nu=0}^{2} b_\nu f(t_{j+\nu}, u^{j+\nu}),\ j = 0, 1, \ldots, M-2,$$

so dass die maximale Konsistenzordnung $p = 4$ vorliegt.

b) Vergleichen Sie die erhaltene Formel mit der im Skript angegebenen Formeltabelle.

c) Diskutieren Sie, ob sich dieses Verfahren und seine Ordnung auch aus Satz 3.1, Kap. III gewinnen lässt.

(6 Punkte)

Aufgabe 26: [Prädiktor-Korrektor Verfahren]

Sei $f \in C^1([t_0, t_E] \times \mathbb{R}^n, \mathbb{R}^n)$ und $\bar{u} \in C^{m+2}([t_0, t_E], \mathbb{R}^n)$ Lösung der Anfangswertaufgabe

$$u' = f(t, u),\ t \in [t_0, t_E],\ u(t_0) = u^0.$$

Zeigen Sie, dass ein Prädiktor–Korrektor Verfahren für diese Aufgabe die Konsistenzordnung $m + 1$ hat, falls ein Adams–Bashforth–Prädiktor der Konsistenzordnung m und ein Adams–Moulton–Korrektor der Konsistenzordnung $m + 1$ verwendet wird. Dabei ist m die Stufe der beiden Mehrschrittverfahren.
Hinweis: Man schreibe das Prädiktor–Korrektor Verfahren wie in der Vorlesung als nichtlineares Mehrschrittverfahren und setze die exakte Lösung ein.

(6 Punkte)

Aufgabe 27: [Lorenz-System mit Mehrschrittverfahren]

Lösen Sie mit den Adams–Bashforth–Verfahren der Ordnung 2 und 4 und der Schrittweite $h = \frac{1}{100}$ das Lorenz-System

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}' = \begin{pmatrix} \sigma(y-x) \\ \lambda x - y - xz \\ -\mu z + xy \end{pmatrix}, \quad \begin{pmatrix} x(0) \\ y(0) \\ z(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 4 \\ 9 \end{pmatrix}$$

für die Parameterwerte $\sigma = 10$, $\lambda = 28$, $\mu = \frac{8}{3}$ und $0 \leqslant t \leqslant 50$.
Verwenden Sie für die Startwerte jeweils ein Einschrittverfahren derselben Ordnung.
Zeichnen Sie die Lösungen in ein (t, x)– bzw. (x, y, z)–Diagramm ein.

(6 Punkte)

Numerik II

Übungsblatt 09

Lösungen:

Aufgabe 25:

Gegeben: Lineares Zweischrittverfahren

$$\frac{1}{h}\sum_{\nu=0}^{2} a_\nu u^{j+\nu} = \sum_{\nu=0}^{2} b_\nu f(t_{j+\nu}, u^{j+\nu}) \quad , \; j=0,\dots,M-2$$

a) Bestimmen Sie die 6 Koeffizienten so, dass Konsistenzordnung $p=4$ vorliegt.
b) Vergleichen Sie die erhaltene Formel mit der im Skript angegebenen Formeltabelle
c) Diskutieren Sie, ob sich dieses Verfahren und seine Ordnung aus Kap III Satz 3.1 gewinnen lässt.

Lösung:

zu a): Setze

$$\rho_K(s) := \frac{s^K}{K!} \quad \Rightarrow \quad \rho_K'(s) = \begin{cases} \frac{s^{K-1}}{(K-1)!} & , K \geq 1 \\ 0 & , K=0 \end{cases}.$$

Nach Kap III §1.2. liegt Konsistenz der Ordnung 4 vor, falls (hier: $p=4, m=2$)

$$\boxed{\sum_{\nu=0}^{2} a_\nu \rho_K(\nu) - \sum_{\nu=0}^{2} b_\nu \rho_K'(\nu) = 0 \;\; \forall K=0,\dots,4 \;\; \& \;\; \sum_{\nu=0}^{2} b_\nu = 1.}$$

Dies liefert uns das folgende Gleichungssystem

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & -1 & -1 & -1 \\ 0 & \frac{1}{2} & 2 & 0 & -1 & -2 \\ 0 & \frac{1}{6} & \frac{4}{3}\left(\frac{8}{6}\right) & 0 & -\frac{1}{2} & -2 \\ 0 & \frac{1}{24} & \frac{2}{3}\left(\frac{16}{24}\right) & 0 & -\frac{1}{6} & -\frac{4}{3}\left(\frac{8}{6}\right) \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

K=0:
K=1:
K=2:
K=3:
K=4:
Normierungs-bedingung:

mit der (eindeutigen) Lösung

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} \\ 0 \\ \frac{1}{2} \\ \frac{1}{6} \\ \frac{2}{3} \\ \frac{1}{6} \end{pmatrix}.$$

zu b): Dies entspricht gerade der Milne-Simpson Mehrstellenformel der Ordnung 4.

zu c): Fall 1: Wenden wir Fall 1 mit

$$m_0 = 0 \;, \; \nu_0 = 0, \; \nu_1 = 2, \; m = 2$$

an, so erhalten wir eine eindeutige Lösung von (3.4), (3.6) mit $\underline{p = \nu_1 - \nu_0 + 1 = 3}$, die durch

$$a_2 = -a_0 = \frac{1}{2} \;, \; a_1 = 0 \;, \; b_0 = \frac{1}{6} = b_2 \;, \; b_1 = \frac{2}{3}$$

gegeben ist. Dies entspricht der Milne-Simpson Formel, jedoch nicht der Konsistenzordnung 4.

Darüberhinaus kann in diesem Fall nur maximal die Konsistenzordnung 3 erreicht werden.

Fall 2: Wenden wir Fall 2 mit

$$u_0 = 0,\ v_0 = 0,\ v_1 = 2,\ u = 2$$

an, so erhalten wir die eindeutige Lösbarkeit von (3.4), (3.6) mit

$$\underline{p = u - u_0 + v_1 - v_0 = 4.}$$

Das Verfahren erhalten wir nun durch Anwenden von Aufgabe 28, die besagt, dass $a_1 = -a_1$, also $a_1 = 0$ gelten muss. Nun lässt sich das GS (3,4), (3,6) mit $p=4$ reduzieren und wegen der Eindeutigkeit, müssen die Koeffizienten durch die aus Fall 1 gegeben sein.

Aufgabe 26:

$$f \in C^1([t_0, t_E] \times \mathbb{R}^n, \mathbb{R}^n)$$

$$\bar{u} \in C^{m+2}([t_0, t_E], \mathbb{R}^n) \text{ Lösung von } u'(t) = f(t, u(t)),\ t \in [t_0, t_E],\ u(t_0) = u^0$$

Zeige:

Adams-Bashforth Prädiktor der Konsistenzordnung m
und
Adams-Moulton Korrektor der Konsistenzordnung $m+1$
} beide m-stufige MSV

$\Rightarrow$ Prädiktor-Korrektor-Verfahren hat Konsistenzordnung $m+1$

Lösung:

1. Prädiktor: Bestimme v

$$\frac{1}{h}\left(\sum_{\nu=0}^{m-1} a_\nu u^{j+\nu} + a_m v\right) = \sum_{\nu=0}^{m-1} b_\nu f(t_{j+\nu}, u^{j+\nu}) \quad , j = 0, \dots, M-m \tag{26.1}$$

(setze $\bar{u}(t_j)$ für u^j und $\bar{u}(t_{j+m})$ für v ein)

Konsistenzfehler (bei t_j):

$$\tau_h^P(t_j) := \frac{1}{h}\sum_{\nu=0}^{m} a_\nu \bar{u}(t_{j+\nu}) - \sum_{\nu=0}^{m-1} b_\nu f(t_{j+\nu}, \bar{u}(t_{j+\nu})) \quad , m \le j \le M-m \tag{26.2}$$

$$\tau_h^P(t_j) = \bar{u}(t_j) - \tilde{u}_j \quad , 0 \le j \le m-1$$

Konsistenz der Ordnung m:

$$\sup_{0 \le j \le M-m} |\tau_h^P(t_j)| \le C_P \cdot h^m. \tag{26.3}$$

2. Korrektor: Bestimme u^{j+m} aus

$$\frac{1}{h}\sum_{\nu=0}^{m} \alpha_\nu u^{j+\nu} = \sum_{\nu=0}^{m-1} \beta_\nu f(t_{j+\nu}, u^{j+\nu}) + \beta_m f(t_{j+m}, v) \quad , j = 0, \dots, M-m \tag{26.4}$$

Konsistenzfehler (bei t_j): (setze $\bar{u}(t_j)$ für u^j und $\bar{u}(t_{j+m})$ für v ein)

$$\tau_h^K(t_j) := \frac{1}{h}\sum_{\nu=0}^{m} \alpha_\nu \bar{u}(t_{j+\nu}) - \sum_{\nu=0}^{m} \beta_\nu f(t_{j+\nu}, \bar{u}(t_{j+\nu})) \quad , m \le j \le M-m \tag{26.5}$$

$$\tau_h^K(t_j) = \bar{u}(t_j) - \tilde{u}_j \quad , 0 \le j \le m-1$$

Konsistenz der Ordnung $m+1$:

$$\sup_{0 \le j \le M-m} |\tau_h^K(t_j)| \le C_K \cdot h^{m+1} \tag{26.6}$$

3. Prädiktor-Korrektor-Verfahren: Löse (26.1) nach v auf

$$v = \frac{1}{a_m}\sum_{\nu=0}^{m-1}\left(h b_\nu f(t_{j+\nu}, u^{j+\nu}) - a_\nu u^{j+\nu}\right)$$

und Setze dies in (26.4) ein

$$\frac{1}{h}\sum_{\nu=0}^{m}\alpha_\nu u^{j+\nu} = \sum_{\nu=0}^{m-1}\beta_\nu f(t_{j+\nu}, u^{j+\nu}) + \beta_m f\Big(t_{j+m}, \frac{1}{a_m}\sum_{\nu=0}^{m-1}\big(h b_\nu f(t_{j+\nu}, u^{j+\nu}) - a_\nu u^{j+\nu}\big)\Big),$$

$$=: \Phi(t_{j+m}, u^j, \dots, u^{j+m}, h)$$

Konsistenzfehler (bei t_j): Aus (26.2) und (26.5) folgt (setze $\bar{u}(t_j)$ für u ein) $j = 0, \dots, M-m$.

$$\tau_n^{PK}(t_j) = \bar{u}(t_j) - \tilde{u}_j \;,\; 0 \le j \le m-1$$

$$\tau_n^{PK}(t_j) := \frac{1}{h}\sum_{\nu=0}^{m}\alpha_\nu \bar{u}(t_{j+\nu}) - \sum_{\nu=0}^{m}\beta_\nu f(t_{j+\nu}, \bar{u}(t_{j+\nu}))$$

(Term ergänzt)

$$+ \beta_m f(t_{j+m}, \bar{u}(t_{j+m})) - \beta_m f\Big(t_{j+m}, \underbrace{\frac{1}{a_m}\sum_{\nu=0}^{m-1}\big(h b_\nu f(t_{j+\nu}, \bar{u}(t_{j+\nu})) - a_\nu \bar{u}(t_{j+\nu})\big)}_{=: w}\Big)$$

(Taylorentw. um w (im 2. Arg.))

$$\overset{(26.2)}{=} \tau_n^{K}(t_j) + \beta_m \cdot \int_0^1 Df(t_{j+m}, w + s(\bar{u}(t_{j+m}) - w))\,ds \cdot (\bar{u}(t_{j+m}) - w)$$

(Satz von Taylor)

$$\overset{(26.5)}{=} \tau_n^{K}(t_j) + \frac{\beta_m}{a_m}\cdot h \cdot \int_0^1 Df(t_{j+m}, w + s(\bar{u}(t_{j+m}) - w))\,ds \cdot \tau_n^{P}(t_j),$$

wobei

$$f(t_{j+m}, \bar{u}(t_{j+m})) = f(t_{j+m}, w) + \int_0^1 D_v f(t_{j+m}, w + s(\bar{u}(t_{j+m}) - w))\,ds \cdot (\bar{u}(t_{j+m}) - w)$$

verwendet wurde, da $f \in C^1$. (!) Dies liefert (für h so klein, dass $|\tau_n^P(t_j)| \le \delta \;\forall j \le M-m$)

$$|\tau_n^{PK}(t_j)| \le |\tau_n^{K}(t_j)| + \Big|\frac{\beta_m}{a_m}\Big| \cdot h \cdot \int_0^1 \underbrace{|D_v f(t_{j+m}, w + s(\bar{u}(t_{j+m}) - w))|}_{\le \sup_{t\in[t_0,t_E]} \sup_{\xi \in B_\delta(\bar{u}(t))} |D_v f(t,\xi)| \le C_f \;(f \in C^1,\ \text{Kompakt})}\,ds \cdot |\tau_n^P(t_j)|$$

$$\overset{(26.3)\,\&\,(26.6)}{\le} C_K \cdot h^{m+1} + \Big|\frac{\beta_m}{a_m}\Big| \cdot h \cdot C_f \cdot C_P \cdot h^m$$

$$= \Big(C_K + \Big|\frac{\beta_m}{a_m}\Big| \cdot C_f \cdot C_P\Big) h^{m+1} =: C_{PK} \cdot h^{m+1}$$

und somit

$$\sup_{0 \le j \le M-m} |\tau_n^{PK}(t_j)| \le C_{PK} \cdot h^{m+1}.$$

```
% Aufgabe 27
%% Initialisierung
sigma = 10;
lambda = 28;
mu = 8/3;
f = @(t,v) [sigma*(v(2)-v(1));lambda*v(1)-v(2)-v(1)*v(3);-mu*v(3)+v(1)*v(2)];
Dvf = @(t,v) [-sigma sigma,0;
              lambda-v(3) -1,-v(1);
              v(2) v(1) -mu];
u_init=[1;4;9];
t_init=0;
t_end=50;
h=1/100;

%% Adams-Bashforth of order 2
L=1;
M=2;
a=[0 -1 1]/L;
b=[-1 3 0]/M;
[tn_adamsbashforth2,un_adamsbashforth2]=AdamsBashforth(f,Dvf,u_init,t_init,↙
t_end,h,a,b);

%% Adams-Bashforth of order 4
L=1;
M=24;
a=[0 0 0 -1 1]/L;
b=[-9 37 -59 55 0]/M;
[tn_adamsbashforth4,un_adamsbashforth4]=AdamsBashforth(f,Dvf,u_init,t_init,↙
t_end,h,a,b);

%% Graphische Ausgabe
set(0,'DefaultAxesFontSize',15.0);
set(0,'DefaultTextFontSize',15.0);
figure
hold on
plot(tn_adamsbashforth2,un_adamsbashforth2(1,:),'r');
plot(tn_adamsbashforth4,un_adamsbashforth4(1,:),'b');
title('Loesung x(t) mit Adams-Bashforth');
xlabel('t'); ylabel('x(t)');
legend('2nd order','4th order','Location','Best');
hold off

figure
plot3(un_adamsbashforth2(1,:),un_adamsbashforth2(2,:),un_adamsbashforth2↙
(3,:),'r');
title('[x(t),y(t),z(t)]');
xlabel('x(t)'); ylabel('y(t)'); zlabel('z(t)');

figure
plot3(un_adamsbashforth4(1,:),un_adamsbashforth4(2,:),un_adamsbashforth4↙
(3,:),'b');
title('[x(t),y(t),z(t)]');
title('[x(t),y(t),z(t)]');
xlabel('x(t)'); ylabel('y(t)'); zlabel('z(t)');
```

```
function [tn,un] = AdamsBashforth(f,Dvf,u_init,t_init,t_end,h,a,b)
%ADAMSBASHFORTH loest loest Anfangswertprobleme der Form
%                u'(t) = f(t,u(t)), u(t_init) = u_init
%             mit (explizitem) Adams-Bashforth Verfahren (a,b)
%             zur konstanten Zeitschrittweite h auf dem endlichen
%             Zeitintervall [t_init,t_end].

%% Initialisierung
  m=length(a)-1;                    % m-Schrittverfahren
  tn=t_init:h:t_end;                % diskretes Zeitintervall
  time_steps=length(tn);            % Anzahl der Zeitschritte (d.h. M=time_steps)
  space_dim=length(u_init);         % Dimension der DGL, Raumdimension
  un=zeros(space_dim,time_steps); % Speicherreservierung fuer RK-Iteration

%% Berechnung m-Anfangsdaten
%  Zur Berechnung der Anfangsdaten soll ein Einschrittverfahren der selben
%  Ordnung genommen werden.
  p=m;                              % Erwartete Konsistenzordnung
  if p==2
    % Euler-Box-Verfahren, Verfahren aus dem Skript Seite 61 (implizit, m=1,↙
p=2)
    %alpha=[1/2];
    %beta=[1/2];
    %gamma=[1];
    %[tn_ESV,un_ESV]=rk_implicit(f,Dvf,u_init,t_init,t_init+h*(m-1),h,alpha,↙
beta,gamma);
    % Heun-Verfahren (explizit, m=p=2)
    %alpha=[0;1];
    %beta=[0 0;1 0];
    %gamma=[1/2 1/2];
    %[tn_ESV,un_ESV]=rk_explicit(f,u_init,t_init,t_init+h*(m-1),h,alpha,beta,↙
gamma);
    % verbessertes Polygonzugverfahren (explizit, m=p=2)
    alpha=[0;1/2];
    beta=[0 0; 1/2 0];
    gamma=[0 1];
    [tn_ESV,un_ESV]=rk_explicit(f,u_init,t_init,t_init+h*(m-1),h,alpha,beta,↙
gamma);
  elseif p==4
    % Klassisches Runge-Kutta-Verfahren (explizit, m=p=4)
    alpha=[0 1/2 1/2 1];
    beta=[0 0 0 0;1/2 0 0 0;0 1/2 0 0;0 0 1 0];
    gamma=[1/6 1/3 1/3 1/6];
    [tn_ESV,un_ESV]=rk_explicit(f,u_init,t_init,t_init+h*(m-1),h,alpha,beta,↙
gamma);
    % Verfahren aus dem Skript Seite 61 (implizit, m=2, p=4)
    %alpha=[(3-sqrt(3))/sqrt(6);(3+sqrt(3))/sqrt(6)];
    %beta=[1/4 1/4-sqrt(3)/6;1/4+sqrt(3)/6 1/4];
    %gamma=[1/2 1/2];
    %[tn_ESV,un_ESV]=rk_implicit(f,Dvf,u_init,t_init,t_init+h*(m-1),h,alpha,↙
beta,gamma);
  else
    error(['Kein Einschrittverfahren der Ordnung p=',num2str(p),'↙
implementiert']);
  end
  un(:,1:m)=un_ESV;
```

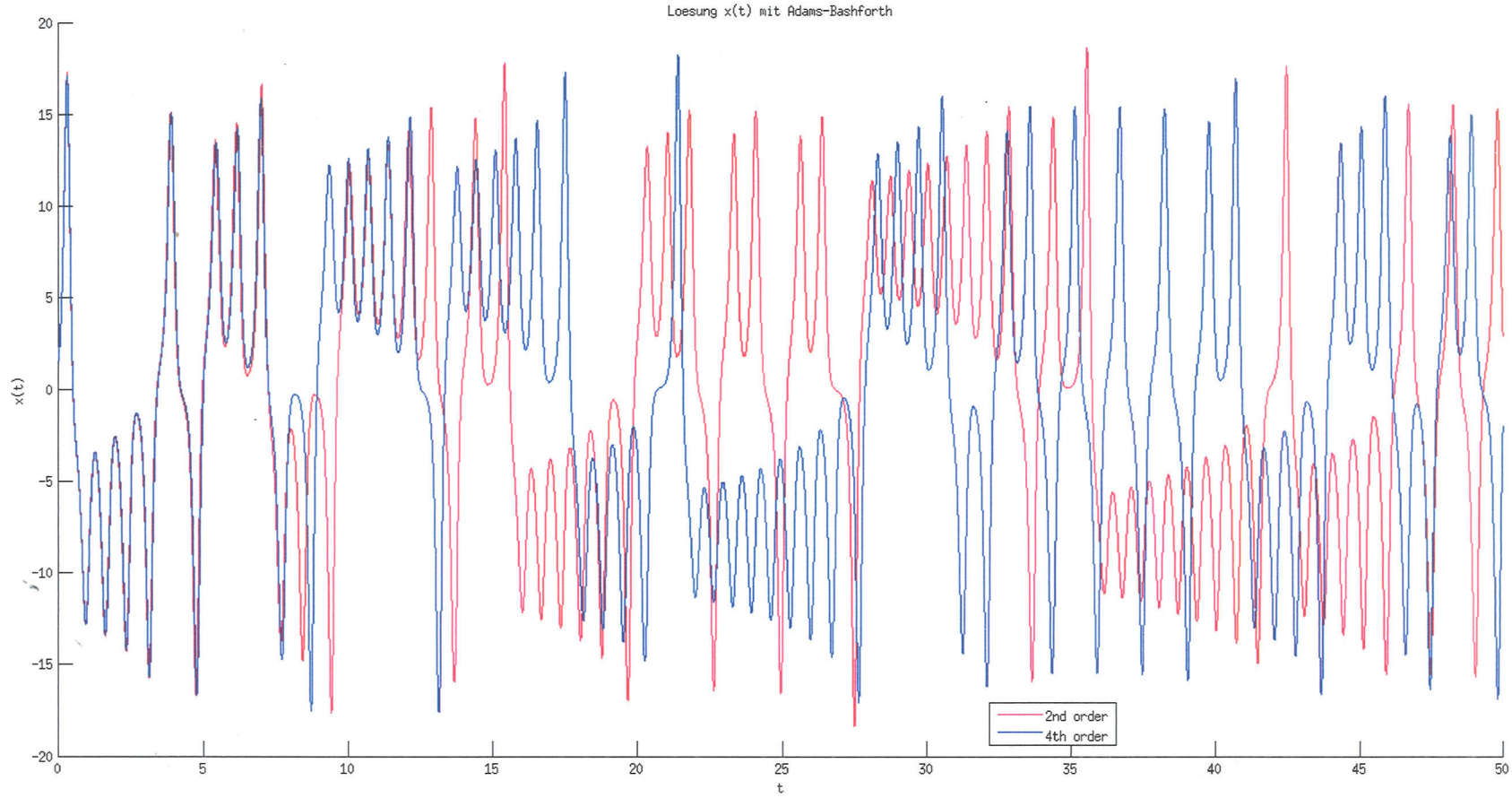
```
  %% Berechnung der Adams-Bashforth Iteration
  M=time_steps;
  for j=0:M-m-1
    sum_temp=zeros(space_dim,1);
    for nu=1:m
        sum_temp=sum_temp-a(nu)*un(:,j+nu)+h*b(nu)*f(tn(j+nu),un(:,j+nu));
    end
    un(:,j+m+1)=sum_temp/a(m+1);
  end
end
```

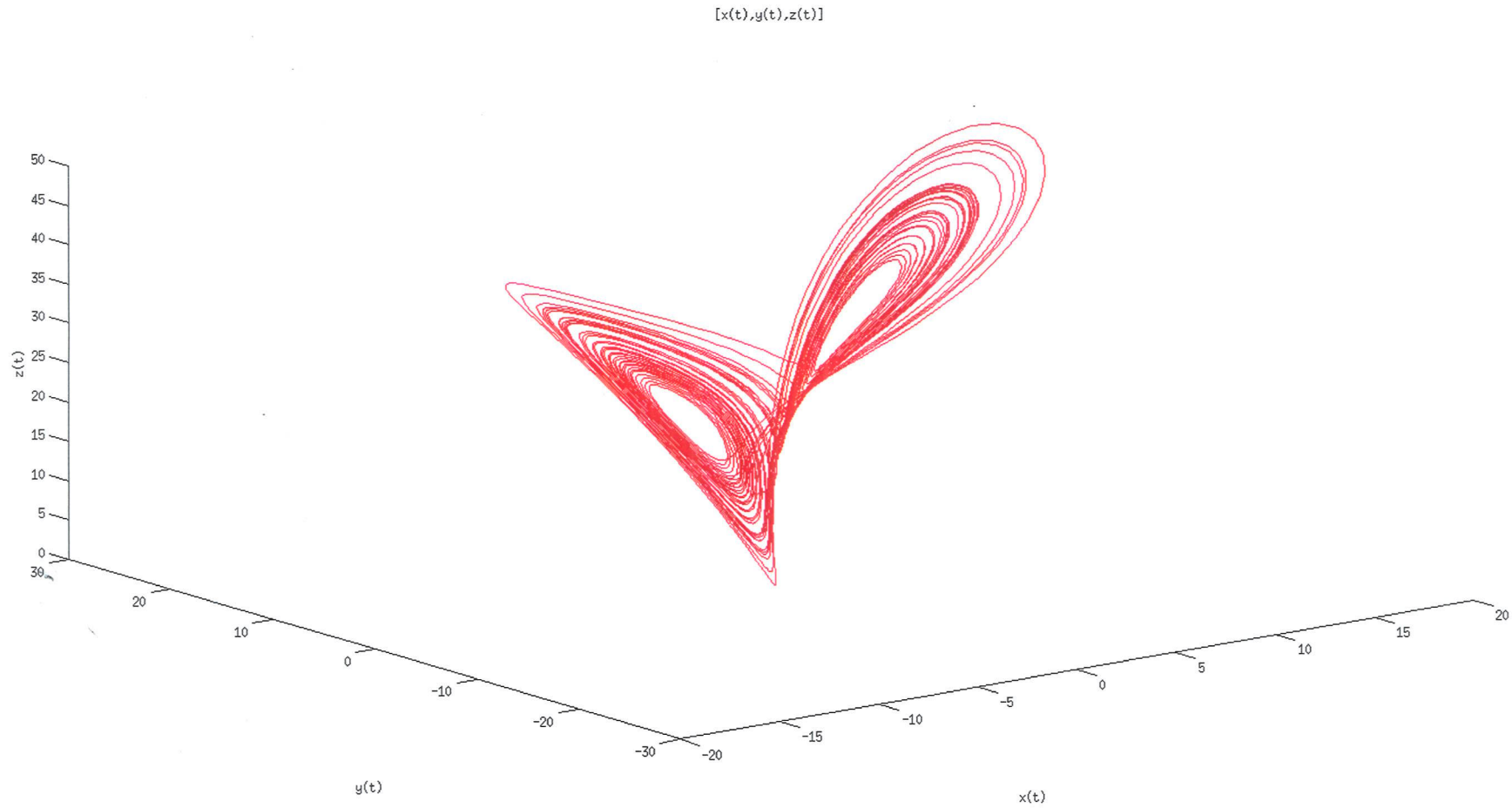
```
function [tn,un] = rk_explicit(f,u_init,t_init,t_end,h,alpha,beta,gamma)
%RK_EXPLICIT loest Anfangswertprobleme der Form
%              u'(t) = f(t,u(t)), u(t_init) = u_init
%            mit expliziten Runge-Kutta Verfahren (alpha,beta,gamma)
%            zur konstanten Zeitschrittweite h auf dem endlichen
%            Zeitintervall [t_init,t_end].

%% Initialisierung
  m=length(gamma);                  % Stufe des RK-Verfahrens
  tn=t_init:h:t_end;                % diskretes Zeitintervall
  time_steps=length(tn);            % Anzahl der Zeitschritte
  space_dim=length(u_init);         % Dimension der DGL, Raumdimension
  un=zeros(space_dim,time_steps); % Speicherreservierung fuer RK-Iteration
  un(:,1)=u_init;                   % Initialisierung der Anfangsdaten
  k=zeros(space_dim,m);             % Speicherreservierung der k_i's

%% Explizites Runge-Kutta-Verfahren
  for j=2:time_steps
    % 1. Rekursive Berechnung der k_i's
    for i=1:m
      temp_k=zeros(space_dim,1);
      for l=1:i-1
        temp_k=temp_k+beta(i,l)*k(:,l);
      end
      k(:,i)=f(tn(j-1)+alpha(i)*h,un(:,j-1)+h*temp_k);
    end

    % 2. Berechnung der u^j's
    temp_phi=zeros(space_dim,1);
    for i=1:m
      temp_phi=temp_phi+gamma(i)*k(:,i);
    end
    un(:,j)=un(:,j-1)+h*temp_phi;
  end
end
```

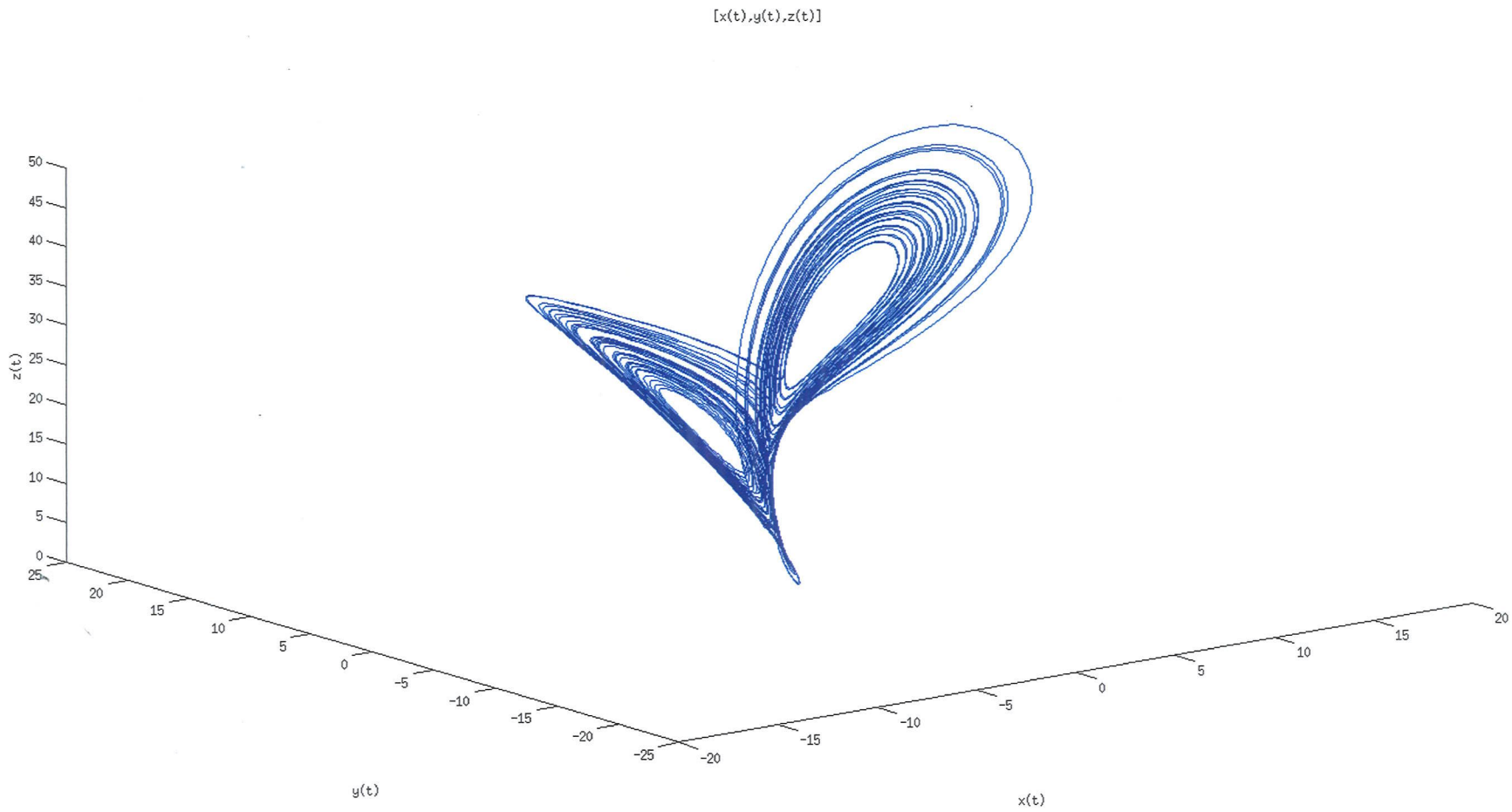
```
function [tn,un] = rk_implicit(f,Dvf,u_init,t_init,t_end,h,alpha,beta,gamma)
%RK_EXPLICIT loest Anfangswertprobleme der Form
%              u'(t) = f(t,u(t)), u(t_init) = u_init
%            mit expliziten Runge-Kutta Verfahren (alpha,beta,gamma)
%            zur konstanten Zeitschrittweite h auf dem endlichen
%            Zeitintervall [t_init,t_end].

%% Initialisierung
  m=length(gamma);                  % Stufe des RK-Verfahrens
  tn=t_init:h:t_end;                % diskretes Zeitintervall
  time_steps=length(tn);            % Anzahl der Zeitschritte
  space_dim=length(u_init);         % Dimension der DGL, Raumdimension
  un=zeros(space_dim,time_steps);   % Speicherreservierung fuer RK-Iteration
  un(:,1)=u_init;                   % Initialisierung der Anfangsdaten
  k=zeros(space_dim,m);             % Speicherreservierung der k_i's

%% Implizites Runge-Kutta-Verfahren
  for j=2:time_steps

    % 1. Berechnung der k_i's durch Loesen eines nichtlinearen
    %    Gleichungssystems
    % Startwert fuer das Newton-Verfahren: (im j-ten Schritt
    % k_i^0=f(t_j,u_j) fuer i=1..m)
    for i=1:m
        k(:,i)=f(tn(j-1),un(:,j-1));
    end
    % Mehrdimensionales Newton-Verahren
    eps=10^(-7);
    max_newton_steps=20;
    n = 0;
    xn = k(:);
    yn = F(f,alpha,beta,xn,tn(j-1),un(:,j-1),h);
    while max(abs(yn))>eps
      n = n+1;
      zn = xn - DF(Dvf,alpha,beta,xn,tn(j-1),un(:,j-1),h)\yn;
      if norm(zn-xn)==Inf
        error('Newton method has not converged: Overflow.')
      end
      xn=zn;
      yn = F(f,alpha,beta,xn,tn(j-1),un(:,j-1),h);
      if n>max_newton_steps
        error('Newton method has not converged: Maximal number of iterations
reached.')
      end
    end
    k(:)=xn;

    % 2. Berechnung der u^j's
    temp_phi=zeros(space_dim,1);
    for i=1:m
      temp_phi=temp_phi+gamma(i)*k(:,i);
    end
    un(:,j)=un(:,j-1)+h*temp_phi;
  end
end
```

```
%% Function F
  function z = F(f,alpha,beta,k,t,v,s)
    d=length(v);
    m=length(alpha);
    z=zeros(d*m,1);
    for i=1:m
      % Berechnung der Argumente von f
      temp_phi=zeros(d,1);
      for j=1:m
        temp_phi=temp_phi+beta(i,j)*k((j-1)*d+1:j*d);
      end
      % Berechnung von F_i
      z((i-1)*d+1:i*d,1)=f(t+alpha(i)*s,v+s*temp_phi)-k((i-1)*d+1:i*d);
    end
  end

%% Function DF
  function A = DF(Df,alpha,beta,k,t,v,s)
    m=length(alpha);
    d=length(v);
    A=zeros(d*m,d*m);
    for i=1:m
      % Berechnung der Argumente von Df
      temp_phi=zeros(d,1);
      for j=1:m
        temp_phi=temp_phi+beta(i,j)*k((j-1)*d+1:j*d);
      end
      % Berechnung von DF_ij
      for j=1:m
        if i==j
          A((i-1)*d+1:i*d,(j-1)*d+1:j*d)=Df(t+alpha(i)*s,v+s*temp_phi)*s*beta
(i,j)-eye(d,d);
        else
          A((i-1)*d+1:i*d,(j-1)*d+1:j*d)=Df(t+alpha(i)*s,v+s*temp_phi)*s*beta
(i,j);
        end
      end
    end
  end

%% Newton-Verfahren:
    function [nst,n]=newton(F,DF,x0,eps,newton_einfach)
    %NEWTON1D Eindimensionales Newton-Verfahren zur Berechnung von Nullstellen
    %         einer Funktion F
    %    F                : Funktion
    %    dF               : Ableitung der Funktion
    %    x0               : Startwert
    %    eps              : Genauigkeit
    %    newton_einfach   : 0 = Newton-Verfahren
    %                       1 = vereinfachtes Newton-Verfahren
    %    nst              : Nullstelle der Funktion
    %    n                : Anzahl der benoetigten Iterationen

        n = 0;
        xn = x0;
        yn = F(xn);
```

```
    if newton_einfach
        A0=DF(x0);
        while max(abs(yn))>eps
            n = n+1;
            %xn = xn - GSV(A0,yn,10^(-10));
            %xn = xn - LR(A0,yn,1);
            xn = xn - A0\yn;
            yn = F(xn);
        end
    else
        while max(abs(yn))>eps
            n = n+1;
            %xn = xn - GSV(dF(xn),yn,10^(-10));
            %xn = xn - LR(dF(xn),yn,1);
            xn = xn - DF(xn)\yn;
            yn = F(xn);
        end
    end
    nst=xn;
end
```


Loesung x(t) mit Adams-Bashforth
x(t)
t
2nd order
4th order
20
15
10
5
0
-5
-10
-15
-20
0
5
10
15
20
25
30
35
40
45
50


[x(t),y(t),z(t)]
z(t)
50
45
40
35
30
25
20
15
10
5
0
30
20
10
0
-10
-20
-30
y(t)
-20
-15
-10
-5
0
5
10
15
20
x(t)


[x(t),y(t),z(t)]
z(t)
50
45
40
35
30
25
20
15
10
5
0
25
20
15
10
5
0
-5
-10
-15
-20
-25
y(t)
-20
-15
-10
-5
0
5
10
15
20
x(t)