Angewandte Numerik 2

Abgabetermin: Freitag 15.11.2013, vor der Übung

Aufgabe 7 (Programmieraufgabe, modifiziertes Euler-Verfahren, Runge-Kutta-Verfahren) (10 Punkte)

Lösen Sie die Anfangswertaufgabe aus Aufgabe 5

\[ y' = -200ty^2, \quad y(-1) = \frac{1}{101} \]

mit den folgenden Einschrittverfahren mit konstanter Schrittweite \( h \). Setze \( n = 1/h \):

\[
\begin{align*}
    t_0 &= -1, \quad y_0 = \frac{1}{101}, \\
    t_{j+1} &= t_j + h, \quad y_{j+1} = y_j + h\phi(t_j, y_j, h), \quad j = 0, \ldots, n-1
\end{align*}
\]

a) modifiziertes Euler-Verfahren: \( \phi(t, y, h) = f(t + h/2, y + h/2f(t, y)) \)

b) Runge-Kutta Verfahren \( \phi(t, y, h) = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \) mit

\[
\begin{align*}
    k_1 &= f(t, y), \\
    k_2 &= f(t + h/2, y + hk_1/2), \\
    k_3 &= f(t + h/2, y + hk_2/2), \\
    k_4 &= f(t + h, y + hk_3).
\end{align*}
\]

Erweitern Sie Ihre Grafik aus Aufgabe 5 um die entsprechenden Daten der beiden Verfahren.

Lösung:

```matlab
function y = modeuler (f, t, y, h)

    % Verfahrensfunktion Modifiziertes Euler-Verfahren
    y = feval (f, t+h/2, y+h/2*feval (f, t, y));

end function

function y = rungekutta(f, t, y, h)

    % Verfahrensfunktion Runge-Kutta-Verfahren
    th2 = t + h/2;
    k1 = feval (f, t, y);
    k2 = feval (f, th2, y+h*k1/2);
    k3 = feval (f, th2, y+h*k2/2);
    k4 = feval (f, t+h, y+h*k3);
    y = 1/6*(k1+2*k2+2*k3+k4);

end function

% Skript zum Erstellen der Grafik

clear all;
close all;
t_0 = -1;
y_0 = 1/101;
```
Aufgabe 8 (Programmieraufgabe, Runge-Kutta-Verfahren mit Schrittweitensteuerung) (12 Punkte)
Betrachten Sie das restringierte Drei-Körper-Problem in Beispiel 37 aus dem Vorlesungsskript. Um die periodischen Bewegungen eines Satelliten im Kraftfeld von Erde und Mond zu berechnen, wird dabei folgendes Szenario angenommen:
Erde und Mond bewegen sich (fast) auf Kreisbahnen um ihren gemeinsamen Schwerpunkt. Ihr Abstand ändert sich dabei nicht und sei hier auf 1 normiert. Weiter sei $\mu = 1/82.45$ die relative Mondmasse und $(1 - \mu)$ die
relative Erdmasse. Die Masse des Satelliten sei im Verhältnis zur Erde- und Mondmasse so klein, dass sie die Bewegung dieser beiden Körper nicht beeinflusst. Ausserdem verlaufe die Bewegung der drei Körper in einer Ebene. Daher lässt sich die Bewegung in dieser Ebene in einem mitrotierenden \((u,v)\)-Koordinatensystem mit Zentrum im Schwerpunkt von Erde und Mond beschreiben. Bei geeigneter Längenskalierung befindet sich dann die Erde im Punkt \((\mu, 0)\) und der Mond im Punkt \((1 - \mu, 0)\).

Die Bewegung des Satelliten in diesem Koordinatensystem wird durch das folgende ODE System 2. Ordnung beschrieben:

\[
\begin{align*}
u'' &= 2v' - \frac{\partial}{\partial u}V, \\
\nu'' &= -2u' - \frac{\partial}{\partial v}V,
\end{align*}
\]

mit der Potentialfunktion

\[
V(u,v) = -\frac{1}{2}(u^2 + v^2) - \frac{1 - \mu}{\sqrt{(u + \mu)^2 + v^2}} - \frac{\mu}{\sqrt{(u - 1 + \mu)^2 + v^2}}
\]

Auf der Homepage finden Sie die Matlab Funktion adaptiv_np aus dem Vorlesungsskript. Schreiben Sie diese zu einer Funktion adaptiv_rkf34 um, die zur numerischen Lösung von Anfangswertaufgaben das eingebettete Runge-Kutta-Fehlberg Verfahren 3. und 4. Ordnung verwendet, welches durch das folgende Butscher-Schema gegeben ist:

\[
\begin{array}{c|cccc}
0 & 1 \\
\frac{1}{3} & 1 & 4 \\
\frac{4}{9} & 4 & 32 & 81 \\
\frac{6}{7} & 57 & -432 & 1053 & 686 \\
1 & 1 & 0 & 27 & 49 & 136 \\
\end{array}
\]

\[
\begin{array}{c|ccccc}
\frac{3}{y_{k+1}} & 1 & 6 & 27 & \frac{49}{52} & 0 \\
\frac{4}{y_{k+1}} & 43 & 288 & 0 & \frac{433}{416} & \frac{343}{1872} & \frac{1}{12} \\
d_{k+1} \approx & -\frac{5}{288} & 0 & \frac{27}{416} & -\frac{245}{1872} & \frac{1}{12}
\end{array}
\]

Lösen Sie das restringierte Drei-Körper-Problem mit dieser Funktion. Testen Sie ihr Programm mit den folgenden zwei Funktionsaufrufen:

\[
\begin{align*}
\text{adaptiv_rkf34}(6.1, 0, [1.2; 0; 0; -1.049357509830350], 0.1, 0.001) \\
\text{adaptiv_rkf34}(17.1, 0, [0.994; 0; 0; -2.0015851], 0.1, 0.001)
\end{align*}
\]

Beachten Sie die unterschiedlichen Masse \((\mu_1 = 1/82.45 \text{ und } \mu_2 = 0.012277471)\). Vergleichen Sie das Ergebnis mit dem des Runge-Kutta-Verfahrens 4. Ordnung mit äquidistanten Schrittweiten. Schreiben Sie hierzu eine Funktion \text{rk4. Hinweis zur Implementierung:} Speichern Sie die Parameter aus dem Butcher-Schema in einer Verfahrensmatrix \((b_{i\ell})\) und einem Knotenvektor \(a\).

\textbf{Lösung:}

Lösungsvariante 1:

\begin{verbatim}
function rk4(tn, t0, y0, h, mu)
% Die Funktion rk4param loest ein parameterabhaengiges AWP
% basierend auf dem 4 stufigen Runge–Kutta–Verfahren von
% mit einer adaptiven Schrittweitensteuerung
\end{verbatim}
Input:  
- \( t_n \): Endzeit
- \( t_0 \): Startzeit
- \( y_0 \): Anfangswerte
- \( h \): Startschrittweite
- \( \mu \): problemabhängiger Parameter

Output: graphische Ausgabe der Lösung

Beispiele für Funktionsaufrufe:
- \( \text{adaptiv_np}(6.1, 0, [1.2; 0; 0; -1.049357509830350], 0.1, 0.001) \) liefert periodische Lösung
- \( \text{adaptiv_np}(17.1, 0, [0.994; 0; 0; -2.0015851], 0.1, 0.001) \) liefert chaotische Lösung

Initialisierung:
- \( y(:,1) = y_0(:,:) \)
- \( t = t_0 \)
- \( T = t \)
- \( H = [] \)
- \( \text{ke} = 0 \)

Verfahrensmatrix:

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & 0 & 0 & 0 & 0 \\
\frac{4}{81} & \frac{32}{81} & 0 & 0 & 0 \\
\frac{57}{98} & -\frac{432}{343} & \frac{1053}{686} & 0 & 0 \\
\frac{1}{6} & 0 & \frac{27}{52} & \frac{49}{156} & 0 \\
\end{bmatrix}
\]

Knotenvektor:
- \( a = [0, 1/4, 4/9, 6/7, 1] \)

Gewichtsvektor:
- \( c = [\frac{43}{288}, 0, \frac{243}{416}, \frac{343}{1872}, \frac{1}{12}] \)

Integrationsschleife:

\[
\text{while and}(t < t_n, \text{ke} < 1000)
\]

\[
\text{flag} = 1;
\]

\[
\text{while flag}
\]

\[
k_1 = f(t, y(:, end), \mu);
\]

\[
w_2 = y(:, end) + h*(b(2,1)*k_1);
\]

\[
k_2 = f(t+a(2)*h, w_2, \mu);
\]

\[
w_3 = y(:, end) + h*(b(3,1)*k_1+b(3,2)*k_2);
\]

\[
k_3 = f(t+a(3)*h, w_3, \mu);
\]

\[
w_4 = y(:, end) + h*(b(4,1)*k_1+b(4,2)*k_2+b(4,3)*k_3);
\]

\[
k_4 = f(t+a(4)*h, w_4, \mu);
\]

\[
w_5 = y(:, end) + h*(b(5,1)*k_1+b(5,2)*k_2+b(5,3)*k_3+b(5,4)*k_4);
\]

\[
k_5 = f(t+a(5)*h, w_5, \mu);
\]
\[ \phi = c(1) \cdot k_1 + c(2) \cdot k_2 + c(3) \cdot k_3 + c(4) \cdot k_4 + c(5) \cdot k_5; \]

end

\[ y = [y, y(:, \text{end}) + h \cdot \phi]; \]
\[ t = t + h; \]
\[ T = [T, t]; \]
\[ H = [H, h]; \]
end

% Grafische Ausgabe
subplot(2,1,1)
semilogy([T(1:end-1); T(2:end)], [H; H], 'g-');
title('Schrittweite');
print -depsc2 adaptiv_vp_fig01

subplot(2,1,2)
plot(y(1,:), y(3,:), 'r-');
title('Phasendiagramm');
print -depsc2 adaptiv_vp_fig02
end

function adaptiv_rk34(tn, t0, y0, h, tol)

% Die Funktion adaptiv_rk34 löst ein AWP basierend auf dem Verfahren von
% Kutta und der verbesserten Polygonzugmethode mit einer adaptiven
% Schrittweitensteuerung

% Input: tn Endzeit
% t0 Startzeit
% y0 Anfangswerte
% h Startschrittweite
% tol Toleranz zu Schrittweitenbestimmung

% Output: graphische Ausgabe der Lösung

% Beispiele für Funktionsaufrufe:
% adaptiv_rk34(6.1, 0, [1.2; 0; 0; -1.049357509830350], 0.1, 0.001)
% -> liefert periodische Lösung
% adaptiv_rk34(17.1, 0, [0.994; 0; 0; -2.0015851], 0.1, 0.001)
% -> liefert chaotische Lösung

% Initialisierung
y(:,1) = y0(:,);
t = t0;
T = t;
H = [];
%w = y(:,1);
ke = 0;

% Verfahrensmatrix
b = [0 0 0 0 0;
     1/4 0 0 0 0;
     4/81 32/81 0 0 0;
     57/98 -432/343 1053/686 0 0;
     1/6 0 27/52 49/156 0;
     1/6 0 27/52 49/156 1/12];

% Knotenvektor
a = [0 1/4 4/9 6/7 1];

% Gewicht für Fehlerschätzer
te = [-5/288 0 27/416 -245/1872 1/12];

% Gewichtsvektoren für rk 3. und 4. Ordnung
gamma = [1/6;0;27/52;49/156;0];
delta = [43/288;0;243/416;343/1872;1/12];

% Integrationsschleife
while and(t<tn, ke < 1000)
    flag = 1;
    while flag
        k1 = f(t,y(:,:));
        w2 = y(:,end) + h*(b(2,1)*k1);
        k2 = f(t+a(2)*h,w2);
        w3 = y(:,end) + h*(b(3,1)*k1+b(3,2)*k2);
        k3 = f(t+a(3)*h,w3);
        w4 = y(:,end) + h*(b(4,1)*k1+b(4,2)*k2+b(4,3)*k3);
        k4 = f(t+a(4)*h,w4);
        w5 = y(:,end) + h*(b(5,1)*k1+b(5,2)*k2+b(5,3)*k3+b(5,4)*k4);
        k5 = f(t+a(5)*h,w5);
        w6 = y(:,end) + h*(b(6,1)*k1+b(6,2)*k2+b(6,3)*k3+b(6,4)*k4+b(6,5)*k5);
        k6 = f(t+h,w6);

        % Berechnung der Verfahrensfunktion
        phi = gamma(1)*k1+gamma(2)*k2+gamma(3)*k3+gamma(4)*k4+gamma(5)*k5;

        % Berechnung des Fehlerschätzers
dk = norm(h*(te(1)*k1+te(3)*k3+te(4)*k4+te(5)*k5));

        % Schrittweitensteuerung
        if dk > 1.2 * tol
            % nicht erfolgreicher Schritt:
            ke = ke+1;
            h = 0.9 * h;
        elseif dk < 0.8 * tol
            % nicht erfolgreicher Schritt:
        end
    end
end
ke = ke+1;

h = 1.1 * h;

else
    % erfolgreicher Schritt:
    ke = ke+1;
    flag = 0;
end

y = [y,y(:,:end)+h*phi];

end

% Grafische Ausgabe

figure(1)
semilogy([T(1:end-1);T(2:end)],[H;H],'g-');
title('Schrittweite');
print -depsc2 adaptiv_vp_fig01

figure(2)
plot(y(1,:),y(3,:),'r-');
title('Phasendiagramm');
print -depsc2 adaptiv_vp_fig02

end

% problemabhaengige Funktion f

function wert = f(t,y)
% Die Funktion f beschreibt das ODE System zu 3-Koerper Problem
% Input:  t
%        y
% Output: wert

mu = 1/82.45; % relative Masse fuer Beispiel 1 hard codiert
mu = 0.012277471 % relative Masse fuer Beispiel 2
z1 = ((y(1)+mu)^2+y(3)^2)^(3/2);
z2 = ((y(1)-1+mu)^2+y(3)^2)^(3/2);
wert = [y(2);
        y(1)+2*y(4)-(1-mu)*(y(1)+mu)/z1-mu*(y(1)-1+mu)/z2;
        y(4);
        y(3)-2*y(2)-(1-mu)*y(3)/z1-mu*y(3)/z2];
close all
clear all

% adaptiv_rk34(6.1,0,[1.2;0;0;−1.049357509830350],0.1,0.001)

adaptiv_rk34(17.1,0,[0.994;0;0;−2.0015851],0.1,0.001)

Abbildung 1: Beispiel 1 (μ = \frac{1}{82.45})

Abbildung 2: Beispiel 2 (μ = 0.012277471)
Lösungsvariante 2 mit parameterabhängigen Funktionen:

```matlab
function [H,T, y] = rk4param(tn , t0 ,y0 ,h,mu)

% Die Funktion rkf4 löst ein parameterabhängiges AWP
% basierend auf dem 4-stufigen Runge-Kutta-Verfahren von
% mit einer adaptiven Schrittweitensteuerung
%
% Input: tn Endzeit
% t0 Startzeit
% y0 Anfangswerte
% h Startschrittweite
% mu problemabhängiger Parameter
%
% Output: H Schrittweitenvektor
% T Zeitschrittvektor
% Y Vektor der Lösungen zu dem Zeitschrittvektor T
%
% Beispiele für Funktionsaufrufe:
% adaptiv_np(6.1 ,0 ,[1.2;0;0;−1.049357509830350],0.1,0.001)
% → liefert periodische Lösung
% adaptiv_np(17.1 ,0 ,[0.994;0;0;−2.0015851],0.1,0.001)
% → liefert chaotische Lösung

% Initialisierung
y(: ,1)=y0 (:);
t = t0 ;
T = t ;
H = [] ;

% Verfahrensmatrix
b = [0 0 0 0 0;
     1/4 0 0 0 0;
     4/81 32/81 0 0 0;
     57/98 −432/343 1053/686 0 0;
     1/6 0 27/52 49/156 0];

% Knotenvektor
a = [0 1/4 4/9 6/7 1];

% Gewichtsvektor
c = [43/288 ,0 ,243/416 ,343/1872 ,1/12];

% Integrationsschleife
while t<tn

    k1 = f(t,y(: ,end), mu);
w2 = y(: ,end) + h*(b(2,1)*k1);
k2 = f(t+a(2)*h, w2, mu);
w3 = y(: ,end) + h*(b(3,1)*k1+b(3,2)*k2);
```
\begin{verbatim}
54 \hspace{1cm} k3 = f(t+a(3)*h,w3, \mu);
55 w4 = y(:,end) + h*(b(4,1)*k1+b(4,2)*k2+b(4,3)*k3);
56 k4 = f(t+a(4)*h, w4, \mu);
57 w5 = y(:,end) + h*(b(5,1)*k1+b(5,2)*k2+b(5,3)*k3+b(5,4)*k4);
58 k5 = f(t+a(5)*h, w5, \mu);
59
60 \% Berechnung der Verfahrensfunktion
61 phi = c(1)*k1+c(2)*k2+c(3)*k3+c(4)*k4+c(5)*k5;
62
63 y = [y,y(:,end)+h*phi];
64 t=t+h;
65 T=[T,t];
66 H=[H,h];
67 end
68 end
69
74 \%—— problemabhaengige Funktion f ———
75
76 function wert = f(t,y,\mu)
77 \% Die Funktion f beschreibt das ODE System zu 3-Koerper Problem
78 \%
79 \% Input: \hspace{1cm} t \hspace{1cm} Zeit
80 \%
81 \% y \hspace{1cm} Vektor
82 \%
83 \% mu \hspace{1cm} relative Masse
84 \%
85 \% Output: \hspace{1cm} wert
86 \%
87 \%
88 \% mu = 1/82.45;
89 z1 = ((y(1)+\mu)^2+y(3)^2)^(3/2);
90 z2 = ((y(1)-1+\mu)^2+y(3)^2)^(3/2);
91 wert = [y(2);
92 y(1)+2*y(4)-(1-\mu)*(y(1)+\mu)/z1-\mu*(y(1)-1+\mu)/z2;
93 y(4);
94 y(3)-2*y(2)-(1-\mu)*y(3)/z1-\mu*y(3)/z2 ];
95 end
\end{verbatim}
% Y Vektor der Lösungen zu dem Zeitschrittenvektor T

% Beispiele für Funktionsaufrufe:

% adaptiv_np(6.1,0,[1.2;0;0;−1.049357509830350],0.1,0.001)
% -> liefert periodische Lösung

% adaptiv_np(17.1,0,[0.994;0;0;−2.0015851],0.1,0.001)
% -> liefert chaotische Lösung

% Initialisierung
y(:,1)=y0(:,);
t = t0;
T = t;
H = [];
ke = 0;

% Verfahrensmatrix
b = [0 0 0 0 0; 1/4 0 0 0 0; 4/81 32/81 0 0 0; 57/98 -432/343 1053/686 0 0; 1/6 0 27/52 49/156 0];

% Knotenvektor
a = [0 1/4 4/9 6/7 1];

% Gewicht für Fehlerschätzer
te = [-5/288 0 27/416 -245/1872 1/12];

% Gewichtsvektoren für rk 3. und 4. Ordnung
gamma = [1/6;0;27/52;49/156;0];

% Integrationsschleife
while and(t<tn , ke < 1000)
flag = 1;

while flag

k1 = f(t,y(:,end), mu);
w2 = y(:,end) + h*(b(2,1)*k1);
k2 = f(t+a(2)*h, w2, mu);
w3 = y(:,end) + h*(b(3,1)*k1+b(3,2)*k2);
k3 = f(t+a(3)*h, w3, mu);
w4 = y(:,end) + h*(b(4,1)*k1+b(4,2)*k2+b(4,3)*k3);
k4 = f(t+a(4)*h, w4, mu);
w5 = y(:,end) + h*(b(5,1)*k1+b(5,2)*k2+b(5,3)*k3+b(5,4)*k4);
k5 = f(t+a(5)*h, w5, mu);

% Berechnung der Verfahrensfunktion
phi = gamma(1)*k1+gamma(2)*k2+gamma(3)*k3+gamma(4)*k4+gamma(5)*k5;

% Berechnung des Fehlerschätzers

\[ dk = \text{norm}(h \cdot (t(1) \cdot k1 + t(3) \cdot k3 + t(4) \cdot k4 + t(5) \cdot k5)); \]

% Schrittweitensteuerung

\[
\text{if } dk > 1.2 \ast \text{tol} \\
\quad \text{ke} = \text{ke}+1; \\
\quad h = 0.9 \ast h; \\
\text{elseif } dk < 0.8 \ast \text{tol} \\
\quad \text{ke} = \text{ke}+1; \\
\quad h = 1.1 \ast h; \\
\text{else} \\
\quad \text{flag} = 0; \\
\end{cases}
\]

end

y = [y,y(:),h*phi];
t = t+h;
T=[T,t];
H=[H,h];
end

% problemabhaengige Funktion f

function wert = f(t,y,mu)
% Die Funktion f beschreibt das ODE System zum 3-Koerper Problem

% Input: t Zeit
% y Vektor
% mu relative Masse
% Output: wert

mu = 1/82.45;
z1 = ((y(1)+mu)^2+y(3)^2)^(3/2);
z2 = ((y(1)-1+mu)^2+y(3)^2)^(3/2);
wert = [y(2);
y(1)+2*y(4)-(1-mu)*(y(1)+mu)/z1-mu*(y(1)-1+mu)/z2;
y(4);
y(3)-2*y(2)-(1-mu)*y(3)/z1-mu*y(3)/z2];
end

% Aufgabe 8. Drei-Koerper Problem

close all
clear all
\texttt{\% Schrittweitenvektor}
\texttt{s = [0.1, 0.01, 0.001];}

\texttt{\% Beispiel 1: periodischer Orbit}

\texttt{figure(1)}

\texttt{for i = 1:3}
\texttt{\% Loese Drei-Koerper Problem mit RKV 4) fuer unterschiedliche Schrittweiten}
\texttt{[H1, T1, y1] = ...}
\texttt{rk4param(6.1,0,[1.2;0;0; -1.049357509830350],s(i),1/82.45);}
\texttt{subplot(4,2,i+(i-1)*1)}
\texttt{semilogy([T1(1:end-1);T1(2:end)], [H1;H1], 'g-');}
\texttt{title(['\texttt{\char24}Schrittweite_{h=\texttt{num2str(s(i))}}']);}
\texttt{print -depsc2 adaptiv_vp_fig01}
\texttt{subplot(4,2,i+i)}
\texttt{plot(y1(1,:), y1(3,:), 'r-');}
\texttt{title('Phasendiagramm:\_Bsp\_1\_mit\_RKV4');}
\texttt{print -depsc2 adaptiv_vp_fig02}
\texttt{end}

\texttt{\% Loese Drei-Koerper Problem mit eingebettetem RKV 3(4) mit Schrittweitensteuerung}
\texttt{[Hla, Tla, yla] = ...}
\texttt{adaptiv_rk34param(6.1,0,[1.2;0;0; -1.049357509830350],0.1,0.001,1/82.45);}
\texttt{subplot(4,2,7)}
\texttt{semilogy([Tla(1:end-1);Tla(2:end)], [Hla;Hla], 'g-');}
\texttt{title('adaptive\_Schrittweite');}
\texttt{print -depsc2 adaptiv_vp_fig01}
\texttt{subplot(4,2,8)}
\texttt{plot(yla(1,:), yla(3,:), 'r-');}
\texttt{title('Phasendiagramm:\_Bsp\_1\_mit\_RKV34');}
\texttt{print -depsc2 adaptiv_vp_fig02}

\texttt{\% Beispiel 2: Chaotischer Orbit}

\texttt{figure(2)}

\texttt{for i = 1:3}
\texttt{\% Loese Drei-Koerper Problem mit RKV 4) fuer unterschiedliche Schrittweiten}
\texttt{[H2, T2, y2] = ...}
\texttt{rk4param(17.1,0,[0.994;0;0; -2.0015851],s(i), 0.012277471);}
\texttt{subplot(4,2,i+(i-1)*1)}
\texttt{end}
semilogy([T2(1:end−1);T2(2:end)],H2;H2,’g–’);

title(['Schrittweite_h=' num2str(s(i))]);

print −depsc2 adaptiv_vp_fig01

subplot(4,2,i+i)
plot(y2(1,:),y2(3,:),’r–’);
title(’Phasendiagramm:Bsp_2.mit_RKV4’);
print −depsc2 adaptiv_vp_fig02

end

% Loese Drei–Koerper Problem mit eingebettetem RKV 3(4) mit
% Schrittweitensteuerung

[H2a, T2a, y2a] = ...
    adaptiv_rk34param(17.1,0,[0.994;0;0;-2.0015851],0.1,0.001, 0.012277471);

subplot(4,2,7)
semilogy([T2a(1:end−1);T2a(2:end)],H2a;H2a,’g–’);
title(’adaptive_Schrittweite’);
print −depsc2 adaptiv_vp_fig01

subplot(4,2,8)
pplot(y2a(1,:),y2a(3,:),’r–’);
ttitle(’Phasendiagramm:Bsp_2.mit_RKV34’);
print −depsc2 adaptiv_vp_fig02

Hinweise:

Die Programmieraufgaben sind in Matlab zu erstellen. Senden Sie alle Files in einer email mit dem Betreff Loesung-Blatt3 an angewandte.numerik@uni-ulm.de (Abgabetermin jeweils wie beim Theorieteil). Drucken Sie zusätzlich allen Programmcode sowie die Ergebnisse aus und geben Sie diese vor der Übung ab. Der Source Code sollte strukturiert und, wenn nötig, dokumentiert sein.
Abbildung 3: Beispiel 1: Vergleich RKV 4 mit verschiedenen Schrittweiten und eingebettetes Runge-Kutta-Fehlberg-Verfahren 3(4) mit adaptiver Schrittweitensteuerung
Abbildung 4: Beispiel 2: Vergleich RKV 4 mit verschiedenen Schrittweiten und eingebettetes Runge-Kutta-Fehlberg-Verfahren 3(4) mit adaptiver Schrittweitensteuerung