MATH 572: Computational Assignment #2

Due on Thurday, March 13, 2014

TTH 12:40pm

Wenqiang Feng

Contents

Adaptive Runge-Kutta Methods Formulas

In this project, we consider two adaptive Runge-Kutta Methods for the following initial-value ODE problem

$$\begin{cases} y'(t) = f(t, y) \\ y(t_0) = y_0, \end{cases} \tag{1}$$

The formula for the fourth order Runge-Kutta (4th RK) method can be read as following

$$\begin{cases} y(t_0) = y_0, \\ K_1 = hf(t_i, y_i) \\ K_2 = hf(t_i + \frac{h}{2}, y_i + \frac{K_1}{2}) \\ K_3 = hf(t_i + \frac{h}{2}, y_i + \frac{K_2}{2}) \\ K_4 = hf(t_i + h, y_i + K_3) \\ y_{i+1} = y_i + \frac{1}{6}(K_1 + K_2 + K_3 + K_4) \end{cases} \tag{2}$$

And the Adaptive Runge-Kutta-Fehlberg (RKF) Method can be wrote as

$$\begin{cases} y(t_0) = y_0, \\ K_1 = hf(t_i, y_i) \\ K_2 = hf(t_i + \frac{h}{4}, y_i + \frac{K_1}{4}) \\ K_3 = hf(t_i + \frac{3h}{8}, y_i + \frac{3}{32}K_1 + \frac{9}{32}K_2) \\ K_4 = hf(t_i + \frac{12h}{13}, y_i + \frac{1932}{2197}K_1 - \frac{7200}{2197}K_2 + \frac{7296}{2197}K_3) \\ K_5 = hf(t_i + h, y_i + \frac{439}{216}K_1 - 8K_2 + \frac{3680}{513}K_3 - \frac{845}{4104}K_4) \\ K_6 = hf(t_i + \frac{h}{2}, y_i - \frac{8}{27}K_1 + 2K_2 - \frac{3544}{2565}K_3 + \frac{1859}{4104}K_4 - \frac{11}{40}) \\ y_{i+1} = y_i + \frac{16}{135}K_1 + \frac{6656}{12825}K_3 + \frac{28561}{56430}K_4 - \frac{9}{50}K_5 + \frac{2}{55}K_6 \\ \tilde{y}_{i+1} = y_i + \frac{25}{216}K_1 + \frac{1408}{2656}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5. \end{cases} \tag{3}$$

The error

$$E = \frac{1}{h}|y_{i+1} - \tilde{y}_{i+1}| \tag{4}$$

will be used as an estimator. If $E \leq Tol$, y will be kept as the current step solution and then move to the next step with time step size δh. If $E > Tol$, recalculate the current step with time step size δh, where

$$\delta = 0.84 \left(\frac{Tol}{E}\right)^{1/4}.$$

Problem 1

1. **The 4th RK method and RKF method for Problem 1.1**

   (a) **Results for Problem 1.1.** From the figure (Fig.1) we can see that the 4th RK method and RKF method are both convergent for Problem 1.1. The 4th RK method is convergent with 4 steps and RKF method with 2 steps and reached error 4.26×10^{-14}.

(b) **Figures** (Fig.1)

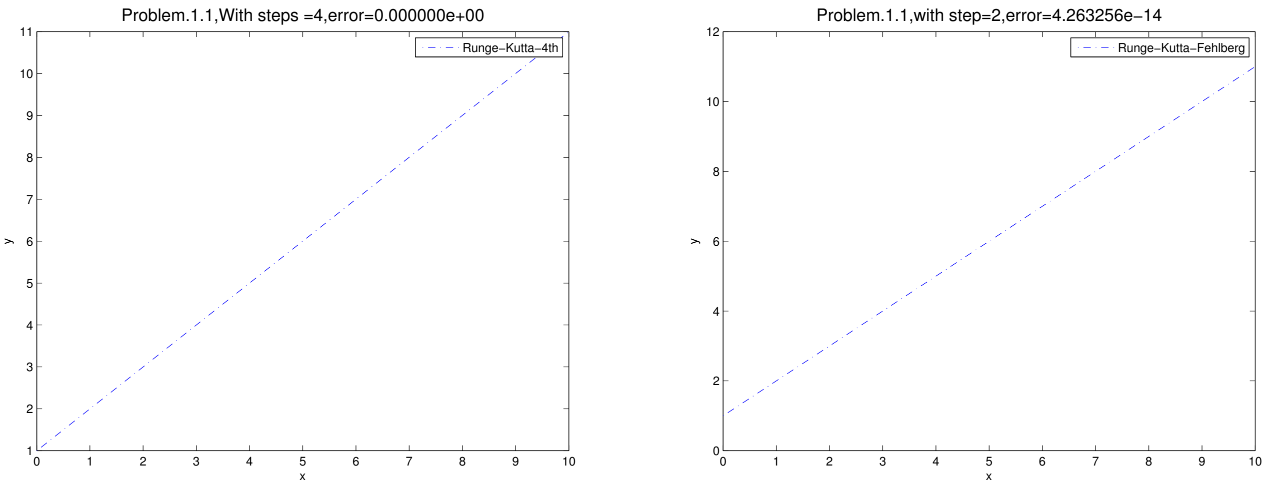

Figure 1: The 4th RK method and RKF method for Problem 1.1

2. **The 4th RK method and RKF method for Problem 1.2**

   (a) **Results for Problem 1.2.** From the figure (Fig.2) we can see that the 4th RK method and RKF method are both convergent for Problem 1.2. The 4th RK method is convergent with 404 steps and reached error 9.9×10^{-6}. RKF method with 29 steps and reached error 2.3×10^{-9}.

   (b) **Figures** (Fig.2)

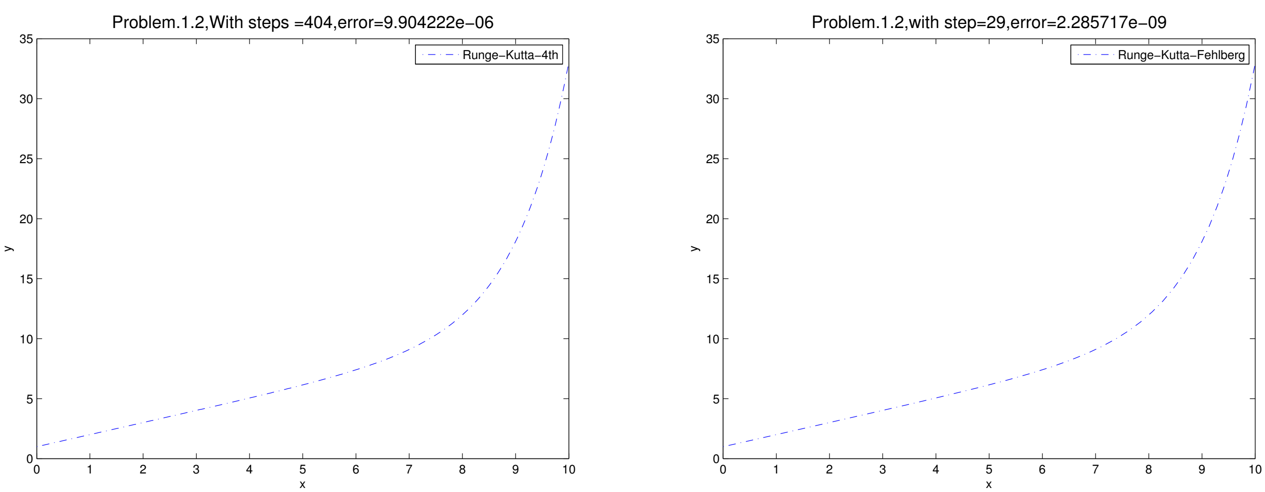

Figure 2: The 4th RK method and RKF method for Problem 1.2

Problem 2

1. **The 4th RK method and RKF method for Problem 2.1**

(a) **Results for Problem 2.1.** From the figure (Fig.3) we can see that the 4th RK method and RKF method are both convergent for Problem 2.1. The 4th RK method is convergent with 24 steps and reached error 7.1×10^{-6}. RKF method with 8 steps and reached error 9.4×10^{-10}.

(b) **Figures** (Fig.3)

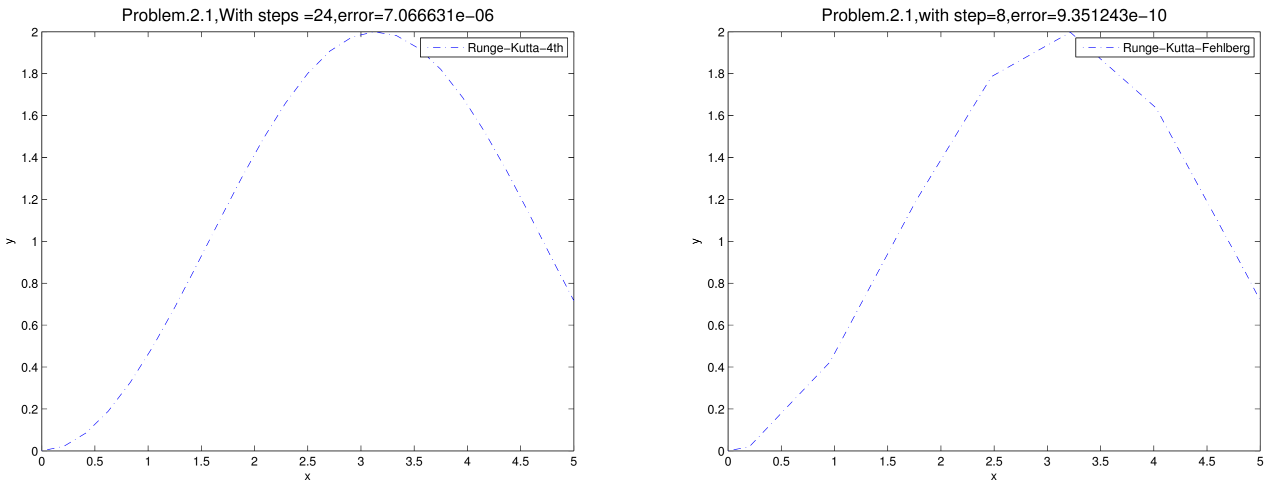

Figure 3: The 4th RK method and RKF method for Problem 2.1

2. **The 4th RK method and RKF method for Problem 2.2**

(a) **Results for Problem 2.2.** From the figure (Fig.4) we can see that the 4th RK method and RKF method are both divergent for Problem 2.2.

(b) **Figures** (Fig.4)

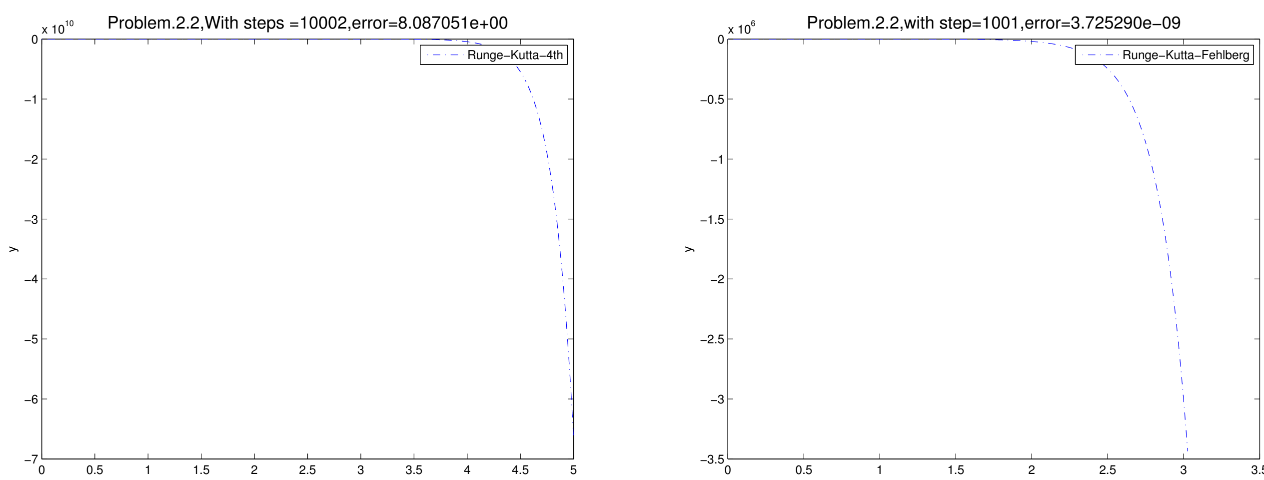

Figure 4: The 4th RK method and RKF method for Problem 2.2

3. **The 4th RK method and RKF method for Problem 2.3**

(a) **Results for Problem 2.3.** From the figure (Fig.5) we can see that the 4th RK method and RKF method are both convergent for Problem 2.3. The 4th RK method is convergent with 96 steps and reached error 9.98×10^{-6}. RKF method with 69 steps and reached error 1.3×10^{-11}.

(b) **Figures** (Fig.5)

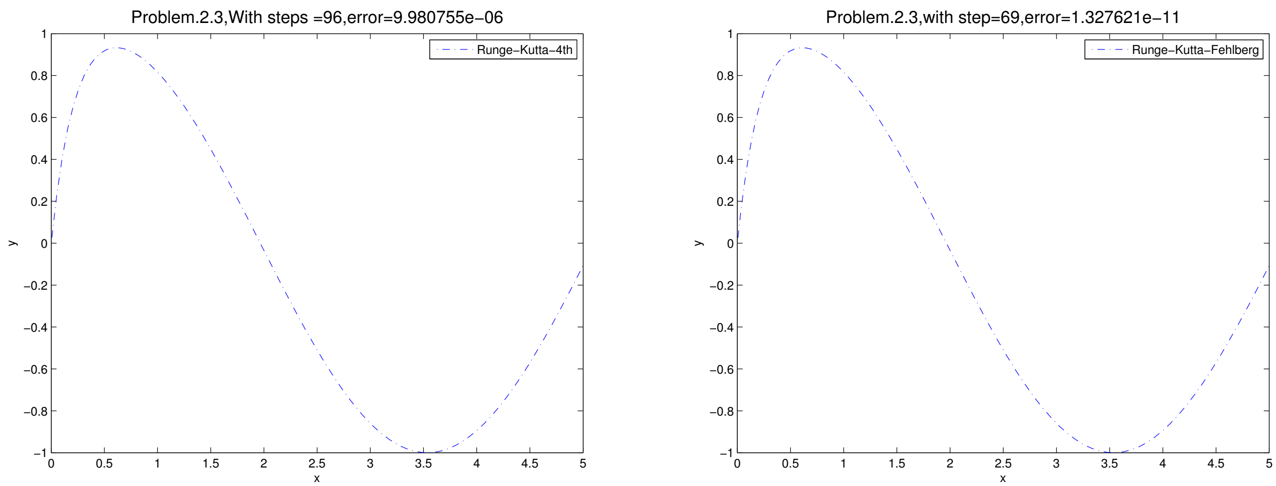

Figure 5: The 4th RK method and RKF method for Problem 2.3

(c) **The 4th RK method and RKF method for Problem 2.4**

i. **Results for Problem 2.4.** From the figure (Fig.6) we can see that the 4th RK method and RKF method are both divergent for Problem 2.4.

ii. **Figures** (Fig.6)

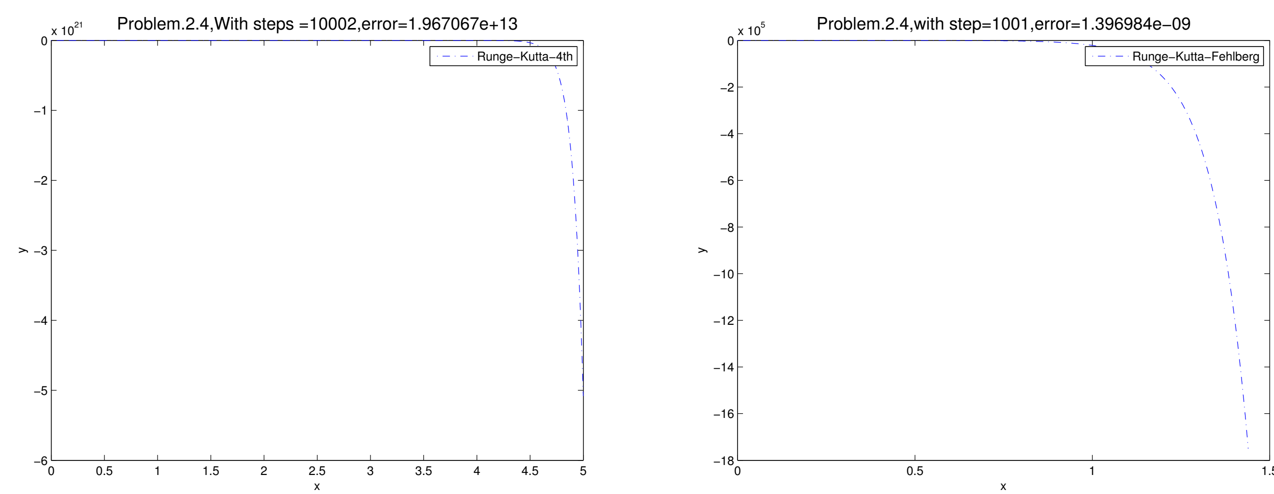

Figure 6: The 4th RK method and RKF method for Problem 2.4

(d) **The 4th RK method and RKF method for Problem 2.5**

i. **Results for Problem 2.5.** From the figure (Fig.7) we can see that the 4th RK method and RKF method are both convergent for Problem 2.5. The 4th RK method is convergent with 88 steps and reached error 8.77×10^{-6}. RKF method with 114 steps and reached error 2.57×10^{-10}.

ii. **Figures** (Fig.7)

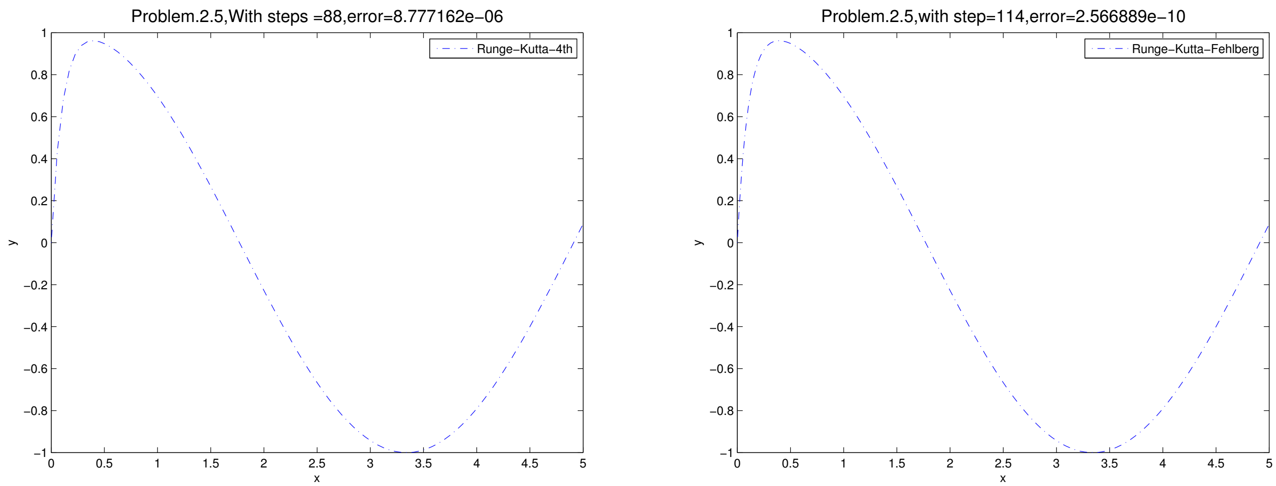

Figure 7: The 4th RK method and RKF method for Problem 2.5

Problem 3

1. **The 4th RK method and RKF method for Problem 3**

   (a) **Results for Problem 3.** From the figure (Fig.8) we can see that the 4th RK method and RKF method are both convergent for Problem 3. The 4th RK method is convergent with 4 steps and reached error 1.77×10^{-15}. RKF method with 2 steps and reached error 2×10^{-15}.

   (b) **Figures** (Fig.8)

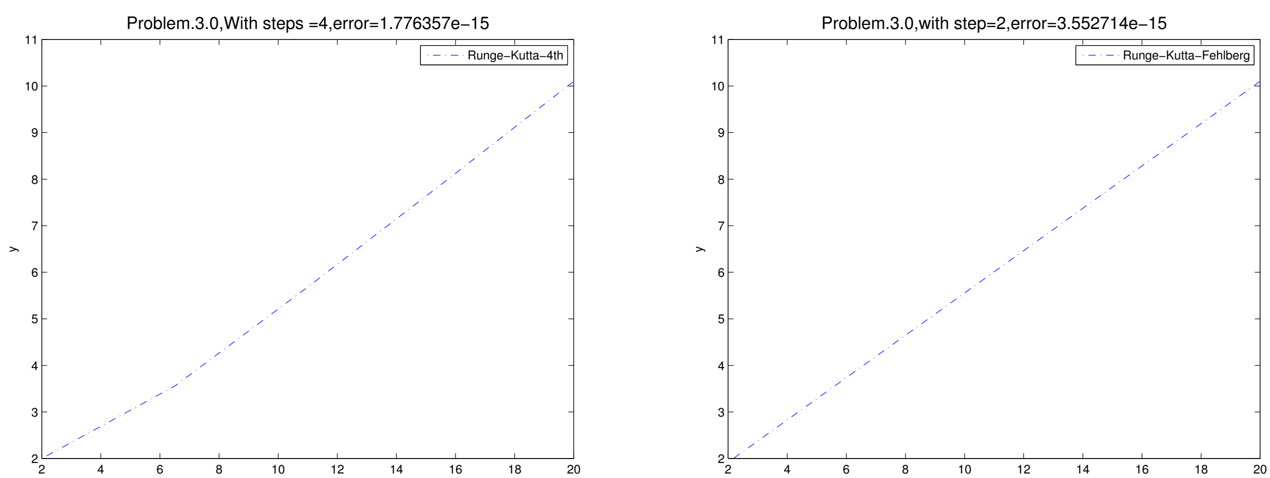

Figure 8: The 4th RK method and RKF method for Problem 3

Problem 4

1. **The 4th RK method and RKF method for Problem 4**

   (a) **Results for Problem 4.** From the figure (Fig.9) we can see that the 4th RK method and RKF method are both convergent for Problem 4. The 4th RK method is convergent with 438 steps and reached error 9.9×10^{-6}. RKF method with 134 steps and reached error 3.68×10^{-14}.

   (b) **Figures** (Fig.9)

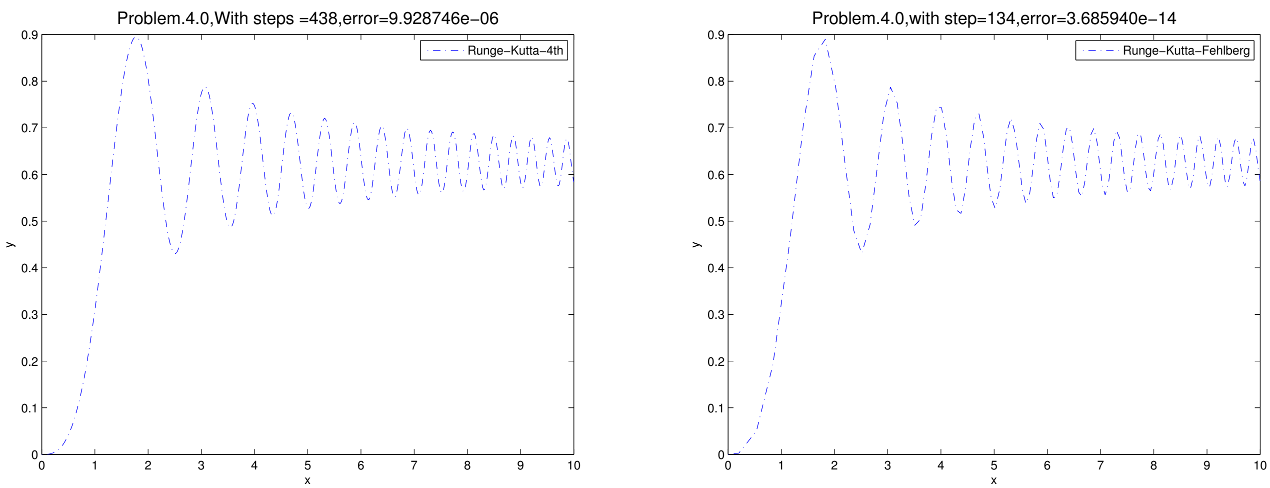

Figure 9: The 4th RK method and RKF method for Problem 4

Problem 5

1. **The 4th RK method and RKF method for Problem 5**

   (a) **Results for Problem 5.** Since, $x = 0$ is the singular point for the problems and $y_0 = \lim_{x \to 0^-} = 1$. So, the schemes do not work for the interval $[-2, 0]$. But schemes works for the interval $[-2, 0-\delta]$ and $\delta > 1 \times 10^{16}$. I changed the problem to the following

$$\begin{aligned} f'(x) &= \tfrac{\ln(1+x)}{x}, x \in [\delta, 2] \\ f(\delta) &= 0. \end{aligned}$$

   The (Fig.8) gives the result for the interval $[\delta, 2]$ and $\delta = 1 \times 10^{10}$.

   (b) **Figures** (Fig.10)

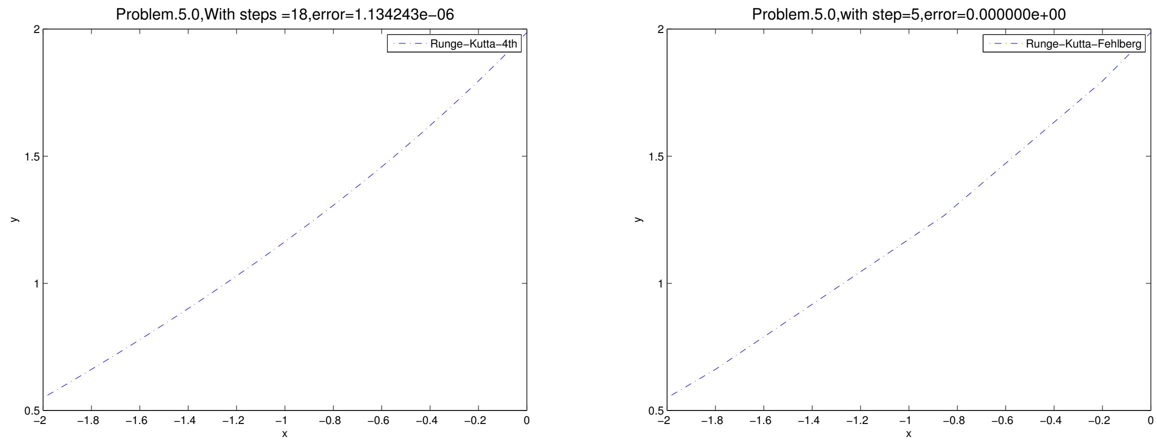

Figure 10: The 4th RK method and RKF method for Problem 5

Adaptive Runge-Kutta Methods MATLAB Code

1. **4-th oder Runge-Kutta Method MATLAB code**

Listing 1: 4-th oder Runge-Kutta Method

```
function [x,y,h]=Runge_Kutta_4(f,xinit,yinit,xfinal,n)
% Euler approximation for ODE initial value problem
% Runge-Kutta 4th order method
% author:Wenqiang Feng
% Email: fw253@mst.edu
% date:January 22, 2012
% Calculation of h from xinit, xfinal, and n
h=(xfinal-xinit)/n;
x=[xinit zeros(1,n)]; y=[yinit zeros(1,n)];

for i=1:n   %calculation loop
x(i+1)=x(i)+h;
k_1 = f(x(i),y(i));
k_2 = f(x(i)+0.5*h,y(i)+0.5*h*k_1);
k_3 = f((x(i)+0.5*h),(y(i)+0.5*h*k_2));
k_4 = f((x(i)+h),(y(i)+k_3*h));

y(i+1) = y(i) + (1/6)*(k_1+2*k_2+2*k_3+k_4)*h;  %main equation
end
```

2. **Main function for problems**

Listing 2: Main function for problem1-5 with 4-th oder Runge-Kutta Method

```
% Script file: main1.m
% The RHS of the differential equation is defined as
% a handle function
% author:Wenqiang Feng
% Email: wfeng1@utk.edu
% date: Mar 8, 2014
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% common parameters
clc
clear all
n=1;
tol=1e-5;
choice=5;  % The choice of the problem number
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% the parameters for each problems
switch choice
case 1.1
% problem 11
f=@(x,y) y-x; %The right hand term
xinit=0;
xfinal=10;
yinit=1;%+1e-3;  %The initial condition
case 1.2
% problem 12
```

```
f=@(x,y) y-x; %The right hand term
xinit=0;
xfinal=10;
yinit=1+1e-3;  %The initial condition
case 2.1
% problem 21
lambda=0;
f=@(x,y) lambda*y+sin(x)-lambda*cos(x); %The right hand term
xinit=0;
xfinal=5;
yinit=0;  %The initial condition
case 2.2
% problem 22
lambda=5;
f=@(x,y) lambda*y+sin(x)-lambda*cos(x); %The right hand term
xinit=0;
xfinal=5;
yinit=0;  %The initial condition
case 2.3
% problem 23
lambda=-5;
f=@(x,y) lambda*y+sin(x)-lambda*cos(x); %The right hand term
xinit=0;
xfinal=5;
yinit=0;  %The initial condition
case 2.4
% problem 24
lambda=10;
f=@(x,y) lambda*y+sin(x)-lambda*cos(x); %The right hand term
xinit=0;
xfinal=5;
yinit=0;  %The initial condition
case 2.5
% problem 25
lambda=-10;
f=@(x,y) lambda*y+sin(x)-lambda*cos(x); %The right hand term
xinit=0;
xfinal=5;
yinit=0;  %The initial condition
case 3
% problem 3
f=@(x,y) 1-y/x; %The right hand term
xinit=2;
xfinal=20;
yinit=2;  %The initial condition
case 4
% problem 4
f=@(x,y) sin(x^2); %The right hand term
xinit=0;
xfinal=10;
yinit=0;  %The initial condition

case 5
```

```
% problem 5
f=@(x,y) log(1+x)/x; %The right hand term
xinit=1e-10;
xfinal=2;
yinit=0;  %The initial condition
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% computing the numberical solutions

y0=100*ones(1,n+1);
[x1,y1]=Runge_Kutta_4(f,xinit,yinit,xfinal,n);

% computing the initial error
%en=max(abs(y1-y0));
en=max(abs(y1(end)-y0(end)));
while (en>tol)
n=n+1;
[x1,y1]=Runge_Kutta_4(f,xinit,yinit,xfinal,n);
[x2,y2,h]=Runge_Kutta_4(f,xinit,yinit,xfinal,2*n);
% two method to computing the error
% temp=interp1(x1,y1,x2);
% en=max(abs(temp-y2));
en=max(abs(y1(end)-y2(end)));
if (n>5000)
disp('the partitions excess 1000')
break;
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plot
figure
plot(x2,y2,'-.')
xlabel('x')
ylabel('y')
legend('Runge-Kutta-4th')
title(sprintf('Problem.%1.1f,With steps =%d,error=%1e',choice,2*n,en),...
'FontSize', 14)
```

3. **Adaptive Runge-Kutta-Fehlberg Method MATLAB code**

Listing 3: 4-th oder Runge-Kutta Method

```
function [time,u,i,E]=Runge_Kutta_Fehlberg(t,T,h,y,f,tol)
% author:Wenqiang Feng
% Email: wfeng1@utk.edu
% date: Mar 8, 2014
u0=y; % initial value
t0=t; % initial time
i=0;  % initial counter
while t<T
h = min(h, T-t);
k1 = h*f(t,y);
```

```
k2 = h*f(t+h/4, y+k1/4);
k3 = h*f(t+3*h/8, y+3*k1/32+9*k2/32);
k4 = h*f(t+12*h/13, y+1932*k1/2197-7200*k2/2197+7296*k3/2197);
k5 = h*f(t+h, y+439*k1/216-8*k2+3680*k3/513-845*k4/4104);
k6 = h*f(t+h/2, y-8*k1/27+2*k2-3544*k3/2565+1859*k4/4104-11*k5/40);
y1 = y + 16*k1/135+6656*k3/12825+28561*k4/56430-9*k5/50+2*k6/55;
y2 = y + 25*k1/216+1408*k3/2565+2197*k4/4104-k5/5;
E=abs(y1-y2);
R = E/h;
delta = 0.84*(tol/R)^(1/4);
if E<=tol
t = t+h;
y = y1;
i = i+1;
fprintf('Step %d: t = %6.4f, y = %18.15f\n', i, t, y);
u(i)=y;
time(i)=t;
h = delta*h;
else
h = delta*h;
end
if (i>1000)
disp('the partitions excess 1000')
break;
end
end
time=[t0,time];
u=[u0,u];
```

4. **Main function for problems**

Listing 4: Main function for problem1-5 with Adaptive Runge-Kutta-Fehlberg Method

```
%% main2
clc
clear all
%% common parameters
tol=1e-5;
h = 0.2;
choice=5;  % The choice of the problem number
%% the parameters for each problems
switch choice
case 1.1
% problem 11
f=@(x,y) y-x; %The right hand term
xinit=0;
xfinal=10;
yinit=1;%+1e-3;  %The initial condition
case 1.2
% problem 12
f=@(x,y) y-x; %The right hand term
xinit=0;
xfinal=10;
yinit=1+1e-3;  %The initial condition
```

```
case 2.1
% problem 21
lambda=0;
f=@(x,y) lambda*y+sin(x)-lambda*cos(x); %The right hand term
xinit=0;
xfinal=5;
yinit=0;  %The initial condition
case 2.2
% problem 22
lambda=5;
f=@(x,y) lambda*y+sin(x)-lambda*cos(x); %The right hand term
xinit=0;
xfinal=5;
yinit=0;  %The initial condition
case 2.3
% problem 23
lambda=-5;
f=@(x,y) lambda*y+sin(x)-lambda*cos(x); %The right hand term
xinit=0;
xfinal=5;
yinit=0;  %The initial condition
case 2.4
% problem 24
lambda=10;
f=@(x,y) lambda*y+sin(x)-lambda*cos(x); %The right hand term
xinit=0;
xfinal=5;
yinit=0;  %The initial condition
case 2.5
% problem 25
lambda=-10;
f=@(x,y) lambda*y+sin(x)-lambda*cos(x); %The right hand term
xinit=0;
xfinal=5;
yinit=0;  %The initial condition
case 3
% problem 3
f=@(x,y) 1-y/x; %The right hand term
xinit=2;
xfinal=20;
yinit=2;  %The initial condition
case 4
% problem 4
f=@(x,y) sin(x^2); %The right hand term
xinit=0;
xfinal=10;
yinit=0;  %The initial condition

case 5
% problem 5
f=@(x,y) log(1+x)/x; %The right hand term
xinit=1e-10;
xfinal=2;
```

```
yinit=0;  %The initial condition
end
% xinit = 0;
% xfinal=2;
% yinit = 0.5;
% f=@(t,y) y-t^2+1; %The right hand term

fprintf('Step %d: t = %6.4f, w = %18.15f\n', 0, xinit, yinit);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% computing the numberical solutions
[time,u,step,error]=Runge_Kutta_Fehlberg(xinit,xfinal,h,yinit,f,tol);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plot
figure
plot(time,u,'-.')
xlabel('x')
ylabel('y')
legend('Runge-Kutta-Fehlberg')
title(sprintf('Problem.%1.1f,with step=%d,error=%1e',choice,step,error),...
'FontSize', 14)
```