Numerical Methods II

Project 2

The Runge-Kutta method of 4th order (the Gill formula) for a system of 2 ordinary differential equations.

December 7, 2018

Rafał Ziomek

Group: EA2

Contents

METHOD DESCRIPTION

Runge-Kutta-Gill method

Method I will use in my calculations is one of the 4th order Runge-Kutta methods - the Gill's formula. For ordinary differential equations it is stated as:

$$x(t+h) = x(t) + \frac{1}{6}[F_1 + (2-\sqrt{2})F_2 + (2+\sqrt{2})F_3 + F_4] \tag{1}$$

where:

$$\begin{cases} h \text{ - step size of the method} \\ \mathrm{F}_1 = hf(t,x) \\ \mathrm{F}_2 = hf(t+\frac{1}{2}h, x+\frac{1}{2}F_1) \\ \mathrm{F}_3 = hf(t+\frac{1}{2}h, x+\frac{1}{2}(\sqrt{2}-1)F_1 + \frac{1}{2}(2-\sqrt{2})F_2) \\ \mathrm{F}_4 = hf(t+h, x-\frac{1}{2}\sqrt{2}F_2 + \frac{1}{2}(2+\sqrt{2})F_3) \end{cases}$$

Presented method can be easily extended to system of two ordinary differential equations. Using vector notation, let:

$$X' = \begin{pmatrix} x_1' = f_1(t, x_1, x_2) \\ x_2' = f_2(t, x_1, x_2) \end{pmatrix} = F(t, X)$$

Now, formula (1) can be rewritten as:

$$X(t+h) = X(t) + \frac{1}{6}[F_1 + (2-\sqrt{2})F_2 + (2+\sqrt{2})F_3 + F_4] \tag{2}$$

where:

$$\begin{cases} h \text{ - step size of the method} \\ \mathrm{F}_1 = hF(t,X) \\ \mathrm{F}_2 = hF(t+\frac{1}{2}h, X+\frac{1}{2}F_1) \\ \mathrm{F}_3 = hF(t+\frac{1}{2}h, X+\frac{1}{2}(\sqrt{2}-1)F_1+\frac{1}{2}(2-\sqrt{2})F_2) \\ \mathrm{F}_4 = hF(t+h, X-\frac{1}{2}\sqrt{2}F_2+\frac{1}{2}(2+\sqrt{2})F_3) \end{cases}$$

Algorithm description

In order to implement Runge-Kutta-Gill method for system of two ordinary differential equations, formula (2) is changed into following:

$$\begin{cases} \mathrm{x}_1(t+h) = x_1(t)+\frac{1}{6}[F_1+(2-\sqrt{2})F_2+(2+\sqrt{2})F_3+F_4] \\ \mathrm{x}_2(t+h) = x_2(t)+\frac{1}{6}[G_1+(2-\sqrt{2})G_2+(2+\sqrt{2})G_3+G_4] \end{cases}$$

where:

$$\begin{cases} h \text{ - step size of the method} \\ \mathrm{F}_1 = hf_1(t,x_1,x_2) \\ \mathrm{G}_1 = hf_2(t,x_1,x_2) \\ \mathrm{F}_2 = hf_1(t+\frac{1}{2}h, x_1+\frac{1}{2}F_1, x_2+\frac{1}{2}G_1) \\ \mathrm{G}_2 = hf_2(t+\frac{1}{2}h, x_1+\frac{1}{2}F_1, x_2+\frac{1}{2}G_1) \\ \mathrm{F}_3 = hf_1(t+\frac{1}{2}h, x_1+\frac{1}{2}(\sqrt{2}-1)F_1+\frac{1}{2}(2-\sqrt{2})F_2, x_2+\frac{1}{2}(\sqrt{2}-1)G_1+\frac{1}{2}(2-\sqrt{2})G_2) \\ \mathrm{F}_G = hf_2(t+\frac{1}{2}h, x_1+\frac{1}{2}(\sqrt{2}-1)F_1+\frac{1}{2}(2-\sqrt{2})F_2, x_2+\frac{1}{2}(\sqrt{2}-1)G_1+\frac{1}{2}(2-\sqrt{2})G_2) \\ \mathrm{F}_4 = hf_1(t+h, x_1-\frac{1}{2}\sqrt{2}F_2+\frac{1}{2}(2+\sqrt{2})F_3, x_2-\frac{1}{2}\sqrt{2}G_2+\frac{1}{2}(2+\sqrt{2})G_3) \\ \mathrm{G}_4 = hf_2(t+h, x_1-\frac{1}{2}\sqrt{2}F_2+\frac{1}{2}(2+\sqrt{2})F_3, x_2-\frac{1}{2}\sqrt{2}G_2+\frac{1}{2}(2+\sqrt{2})G_3) \end{cases}$$

My algorithm also returns error of calculated $x_{1,i}, x_{2,i}$, by substracting it from known exact solution.

MATLAB FUNCTIONS CODE

solveODE

```
%Function takes as arguments:
% ODE1 - first ordinary differential equation in the system as anonymous function
% ODE2 - second ordinary differential equation in the system as anonymous function
% a - start of the calculations interval
% b - end of the calculations interval
% n - number of steps
% initODE1 - initial value for ODE1
% initODE2 - initial value for ODE2
% exactODE1 - exact solution function of first ordinary differential as anonymous function
% exactODE2 - exact solution function of second ordinary differential equation in the system as anonymous function

%Function returns:
%t - vector of used variables
%x - calculated values of x(t) - from ODE1
%y - calculated values of y(t) - from ODE2
%errODE1 - error of calculated values of x(t)
%errODE2 - error of calculated values of y(t)

function [t, x, y, errODE1, errODE2]=solveODE(ODE1, ODE2, a, b, n, initODE1, initODE2, exactODE1, exactODE2)
h = (b-a)/n;                 %h - step size
t = a:h:b;                   %initialization of needed vectors
x = zeros(1,n+1);
y = zeros(1,n+1);
errODE1 = zeros(1,n+1);
errODE2 = zeros(1,n+1);
x(1) = initODE1;             %assigning initial values in the vector
```

```
y(1) = initODE2;
sqroot2 = sqrt(2);          %calculation of square root of 2 so it is
    not calculated every time when needed
for i=1:n                   %loop of calculations, described in report
    F1 = h*ODE1(t(i), x(i), y(i));
    G1 = h*ODE2(t(i), x(i), y(i));
    F2 = h*ODE1(t(i)+0.5*h, x(i)+0.5*F1, y(i)+0.5*G1);
    G2 = h*ODE2(t(i)+0.5*h, x(i)+0.5*F1, y(i)+0.5*G1);
    F3 = h*ODE1(t(i)+0.5*h, x(i)+0.5*(sqroot2-1)*F1 + 0.5*(2-sqroot2)*
        F2, y(i) + 0.5*(sqroot2-1)*G1 + 0.5*(2-sqroot2)*G2);
    G3 = h*ODE2(t(i)+0.5*h, x(i)+0.5*(sqroot2-1)*F1 + 0.5*(2-sqroot2)*
        F2, y(i) + 0.5*(sqroot2-1)*G1 + 0.5*(2-sqroot2)*G2);
    F4 = h*ODE1(t(i)+h, x(i)-0.5*sqroot2*F2+0.5*(2+sqroot2)*F3, y(i)
        -0.5*sqroot2*G2+0.5*(2+sqroot2)*G3);
    G4 = h*ODE2(t(i)+h, x(i)-0.5*sqroot2*F2+0.5*(2+sqroot2)*F3, y(i)
        -0.5*sqroot2*G2+0.5*(2+sqroot2)*G3);

    x(i+1) = x(i) + (1/6)*(F1 + (2-sqroot2)*F2 + (2+sqroot2)*F3 + F4);
    y(i+1) = y(i) + (1/6)*(G1 + (2-sqroot2)*G2 + (2+sqroot2)*G3 + G4);
    errODE1(i+1) = exactODE1(t(i+1), x(i+1), y(i+1)) - x(i+1);
    errODE2(i+1) = exactODE2(t(i+1), x(i+1), y(i+1)) - y(i+1);
end
end
```

test

```
%TEST FILE, AFTER EACH PLOT PLEASE PRESS ANY BUTTON TO SEE PLOT FROM
    NEXT TEST
%SYSTEM OF ODE FOR TESTS 1-4

%ODE1: x' = x + 4y - e^t
%ODE2: y' = x + y + 2e^t
%x(0) = 4
%y(0) = 5/4
%exactX: 4e^3t + 2e^-t - 2e^t
%exactY: 2e^3t - e^-t + 1/4e^t

%TEST 1 # of steps:5

n=5;
[t,x,y,errx,erry] = solveODE(@(t,x,y)x+4*y-exp(t), @(t,x,y) x + y + 2*
    exp(t), 0, 1, n, 4, 1.25, @(t,x,y) 4*exp(3*t) + 2*exp(-t) - 2*exp(t)
    , @(t,x,y)2*exp(3*t) - exp(-t) + 0.25*exp(t));

ax1 = subplot(2,1,1);
plot(ax1,t,x, t,y);
title(ax1,'Calculated values')
ylabel(ax1,'f(t)')
xlabel(ax1,'t')
legend('x(t)','y(t)')
ax1 = subplot(2,1,2);
plot(ax1,t,errx,t,erry);
title(ax1,'Errors')
ylabel(ax1,'exact(t) - f(t)')
xlabel(ax1,'t')
legend('error x(t)','error(y(t))')

waitforbuttonpress

```

```
%TEST 2 # of steps:20

n=20;
[t,x,y,errx,erry] = solveODE(@(t,x,y)x+4*y-exp(t), @(t,x,y) x + y + 2*
    exp(t), 0, 1, n, 4, 1.25, @(t,x,y) 4*exp(3*t) + 2*exp(-t) - 2*exp(t)
    , @(t,x,y)2*exp(3*t) - exp(-t) + 0.25*exp(t));

ax1 = subplot(2,1,1);
plot(ax1,t,x, t,y);
title(ax1,'Calculated values')
ylabel(ax1,'f(t)')
xlabel(ax1,'t')
legend('x(t)','y(t)')
ax1 = subplot(2,1,2);
plot(ax1,t,errx,t,erry);
title(ax1,'Errors')
ylabel(ax1,'exact(t) - f(t)')
xlabel(ax1,'t')
legend('error x(t)','error(y(t))')

waitforbuttonpress

%TEST 3 # of steps:100

n=100;
[t,x,y,errx,erry] = solveODE(@(t,x,y)x+4*y-exp(t), @(t,x,y) x + y + 2*
    exp(t), 0, 1, n, 4, 1.25, @(t,x,y) 4*exp(3*t) + 2*exp(-t) - 2*exp(t)
    , @(t,x,y)2*exp(3*t) - exp(-t) + 0.25*exp(t));

ax1 = subplot(2,1,1);
plot(ax1,t,x, t,y);
title(ax1,'Calculated values')
ylabel(ax1,'f(t)')
xlabel(ax1,'t')
legend('x(t)','y(t)')
```

```
ax1 = subplot(2,1,2);
plot(ax1,t,errx,t,erry);
title(ax1,'Errors')
ylabel(ax1,'exact(t) - f(t)')
xlabel(ax1,'t')
legend('error x(t)','error(y(t))')

waitforbuttonpress

%TEST 4 # of steps:1000

n=1000;
[t,x,y,errx,erry] = solveODE(@(t,x,y)x+4*y-exp(t), @(t,x,y) x + y + 2*
    exp(t), 0, 1, n, 4, 1.25, @(t,x,y) 4*exp(3*t) + 2*exp(-t) - 2*exp(t)
    , @(t,x,y)2*exp(3*t) - exp(-t) + 0.25*exp(t));

ax1 = subplot(2,1,1);
plot(ax1,t,x, t,y);
title(ax1,'Calculated values')
ylabel(ax1,'f(t)')
xlabel(ax1,'t')
legend('x(t)','y(t)')
ax1 = subplot(2,1,2);
plot(ax1,t,errx,t,erry);
title(ax1,'Errors')
ylabel(ax1,'exact(t) - f(t)')
xlabel(ax1,'t')
legend('error x(t)','error(y(t))')

waitforbuttonpress

%SYSTEM OF ODE FOR TEST 5-6

%ODE1: x' = x - y
%ODE2: y' = x + 3y
```

```
%exactX: e^2t
%exactY: -e^2t

%TEST 5, n=10

n=10;
a=0;
b=1;
initx = exp(2*a);
inity = -exp(2*a);

[t,x,y,errx,erry] = solveODE(@(t,x,y) x - y, @(t,x,y) x + 3*y, a, b, n,
    initx, inity, @(t,x,y)exp(2*t), @(t,x,y)-exp(2*t));

ax1 = subplot(2,1,1);
plot(ax1,t,x, t,y);
title(ax1,'Calculated values')
ylabel(ax1,'f(t)')
xlabel(ax1,'t')
legend('x(t)','y(t)')
ax1 = subplot(2,1,2);
plot(ax1,t,errx,t,erry);
title(ax1,'Errors')
ylabel(ax1,'exact(t) - f(t)')
xlabel(ax1,'t')
legend('error x(t)','error(y(t))')

waitforbuttonpress

%TEST 6, n=1000

n=1000;
a=0;
b=1;
initx = exp(2*a);
```

```
inity = -exp(2*a);

[t,x,y,errx,erry] = solveODE(@(t,x,y) x - y, @(t,x,y) x + 3*y, a, b, n,
    initx, inity, @(t,x,y)exp(2*t), @(t,x,y)-exp(2*t));

ax1 = subplot(2,1,1);
plot(ax1,t,x, t,y);
title(ax1,'Calculated values')
ylabel(ax1,'f(t)')
xlabel(ax1,'t')
legend('x(t)','y(t)')
ax1 = subplot(2,1,2);
plot(ax1,t,errx,t,erry);
title(ax1,'Errors')
ylabel(ax1,'exact(t) - f(t)')
xlabel(ax1,'t')
legend('error x(t)','error(y(t))')

waitforbuttonpress

%STIFF SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS TESTS 8-10
%ODE1: x' = -80.6x + 119.4y
%ODE2: y' = 79.6x - 120.4y
%exactX: c1*3*e^-t + (-c2) * e^-200t
%exactY: c1*2*e^-t + c2 * e^-200t
%c1 and c2 depend on initial conditions, if c2==0, then solution should
    be
%smooth, otherwise it should produce rapid changes while approaching to
%solution

%TEST 7: c1=5, c2=0, hence x(0) = 15 y(0)= 10
c1 = 5;
c2 = 0;
a = 0;
b = 1;
```

```
initx = c1*3*exp(-a) + (-c2) * exp(-200*a);
inity = c1*2*exp(-a) + c2 * exp(-200*a);
[t,x,y,errx,erry] = solveODE(@(t,x,y)-80.6*x + 119.4*y, @(t,x,y)79.6*x
    - 120.4*y, a, b, 200, initx, inity, @(t,x,y)c1*3*exp(-t) + (-c2) *
    exp(-200*t), @(t,x,y)c1*2*exp(-t) + c2 * exp(-200*t));

ax1 = subplot(2,1,1);
plot(ax1,t,x, t,y);
title(ax1,'Calculated values')
ylabel(ax1,'f(t)')
xlabel(ax1,'t')
legend('x(t)','y(t)')
ax1 = subplot(2,1,2);
plot(ax1,t,errx,t,erry);
title(ax1,'Errors')
ylabel(ax1,'exact(t) - f(t)')
xlabel(ax1,'t')
legend('error x(t)','error(y(t))')

waitforbuttonpress

%TEST 8: c1=5, c2=1, hence x(0) = 14 y(0)= 12
c1 = 5;
c2 = 1;
a = 0;
b = 1;
initx = c1*3*exp(-a) + (-c2) * exp(-200*a);
inity = c1*2*exp(-a) + c2 * exp(-200*a);
[t,x,y,errx,erry] = solveODE(@(t,x,y)-80.6*x + 119.4*y, @(t,x,y)79.6*x
    - 120.4*y, a, b, 200, initx, inity, @(t,x,y)c1*3*exp(-t) + (-c2) *
    exp(-200*t), @(t,x,y)c1*2*exp(-t) + c2 * exp(-200*t));

ax1 = subplot(2,1,1);
plot(ax1,t,x, t,y);
title(ax1,'Calculated values')
```

```
ylabel(ax1,'f(t)')
xlabel(ax1,'t')
legend('x(t)','y(t)')
ax1 = subplot(2,1,2);
plot(ax1,t,errx,t,erry);
title(ax1,'Errors')
ylabel(ax1,'exact(t) - f(t)')
xlabel(ax1,'t')
legend('error x(t)','error(y(t))')

waitforbuttonpress

%TEST 9: c1=1, c2=0, hence x(0) = 3 y(0)= 2
c1 = 1;
c2 = 0;
a = 0;
b = 1;
initx = c1*3*exp(-a) + (-c2) * exp(-200*a);
inity = c1*2*exp(-a) + c2 * exp(-200*a);
[t,x,y,errx,erry] = solveODE(@(t,x,y)-80.6*x + 119.4*y, @(t,x,y)79.6*x
    - 120.4*y, a, b, 200, initx, inity, @(t,x,y)c1*3*exp(-t) + (-c2) *
    exp(-200*t), @(t,x,y)c1*2*exp(-t) + c2 * exp(-200*t));

ax1 = subplot(2,1,1);
p = plot(ax1,t,x, t,y);
title(ax1,'Calculated values')
ylabel(ax1,'f(t)')
xlabel(ax1,'t')
legend('x(t)','y(t)')
ax1 = subplot(2,1,2);
plot(ax1,t,errx,t,erry);
title(ax1,'Errors')
ylabel(ax1,'exact(t) - f(t)')
xlabel(ax1,'t')
legend('error x(t)','error(y(t))')
```

```
waitforbuttonpress

%TEST 10: c1=1, c2=5, hence x(0) = -2 y(0)= 7
c1 = 1;
c2 = 5;
a = 0;
b = 1;
initx = c1*3*exp(-a) + (-c2) * exp(-200*a);
inity = c1*2*exp(-a) + c2 * exp(-200*a);
[t,x,y,errx,erry] = solveODE(@(t,x,y)-80.6*x + 119.4*y, @(t,x,y)79.6*x
    - 120.4*y, a, b, 200, initx, inity, @(t,x,y)c1*3*exp(-t) + (-c2) *
    exp(-200*t), @(t,x,y)c1*2*exp(-t) + c2 * exp(-200*t));

ax1 = subplot(2,1,1);
plot(ax1,t,x, t,y);
title(ax1,'Calculated values')
ylabel(ax1,'f(t)')
xlabel(ax1,'t')
legend('x(t)','y(t)')
ax1 = subplot(2,1,2);
plot(ax1,t,errx,t,erry);
title(ax1,'Errors')
ylabel(ax1,'exact(t) - f(t)')
xlabel(ax1,'t')
legend('error x(t)','error(y(t))')
```

NUMERICAL EXAMPLES AND ANALYSIS

Tests 1-4

For tests 1-4 I used following system of differential equations:

$$\begin{cases} x_1'(t) = x_1 + 4x_2 - e^t \\ x_2'(t) = x_1 + x_2 - 2e^t \end{cases}$$

with initial condition:

$$\begin{cases} x_1(0) = 4 \\ x_2(0) = \frac{5}{4} \end{cases}$$

and exact solution:

$$\begin{cases} x_1(t) = 4e^{3t} + 2e^{-t} - 2e^t \\ x_2(t) = 2e^{3t} - e^{-t} + \frac{1}{4}e^t \end{cases}$$

In each test, the number of steps was increasing. For test 1 - 5 steps, test 2 - 20 steps, test 3 - 100 steps and test 4 - 1000 steps.
On the next pages there are plots of calculated solutions and errors.
As the number of steps increase (hence the step size gets smaller), the error of the method changes in magnitude, as expected. What I found quite interesting, is the fact that plots of and error and the function itself are very similar in shape.

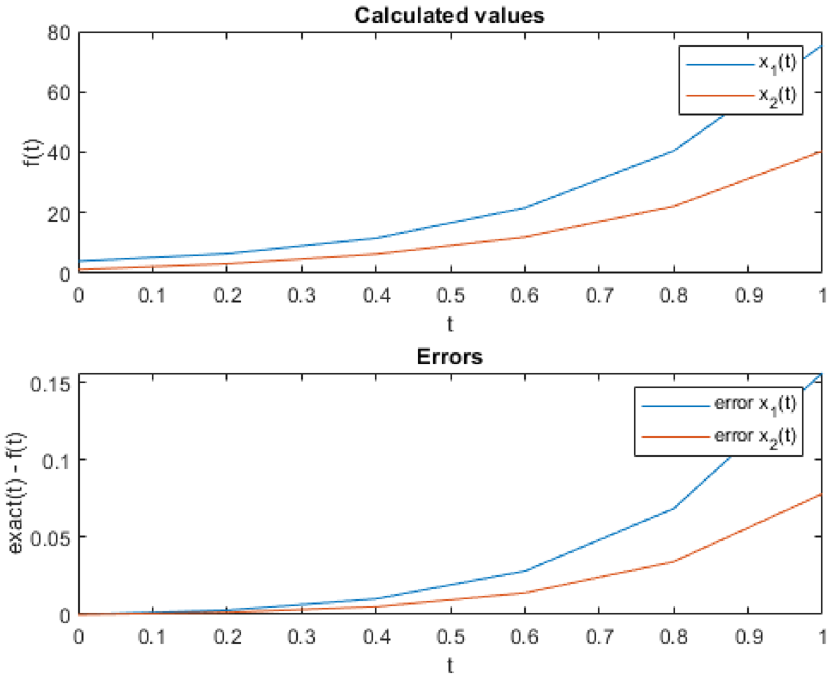

Figure 1: Test 1

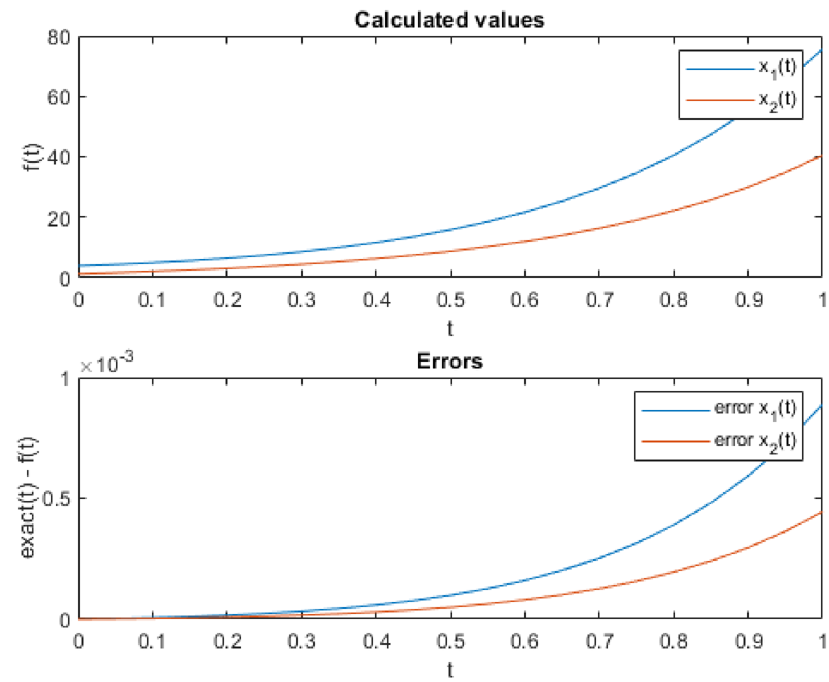

Figure 2: Test 2

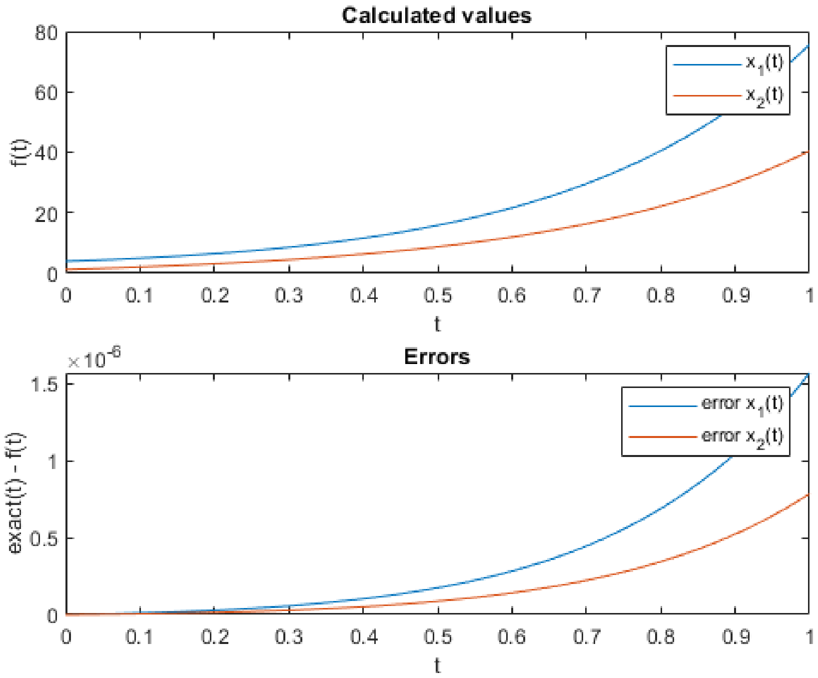

Figure 3: Test 3

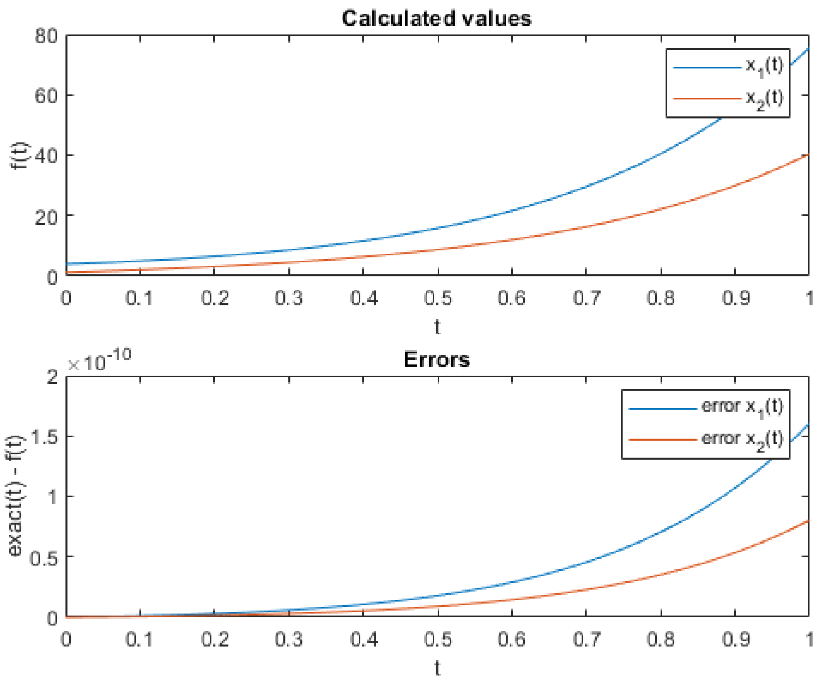

Figure 4: Test 4

Tests 5-6

For tests 5-6 I used following system of differential equations:

$$\begin{cases} x_1'(t) = x_1 - x_2 \\ x_2'(t) = x_1 + 3x_2 \end{cases}$$

with initial condition:

$$\begin{cases} x_1(0) = 2 \\ x_2(0) = -2 \end{cases}$$

and exact solution:

$$\begin{cases} x_1(t) = e^{2t} \\ x_2(t) = -e^{2t} \end{cases}$$

As in previous tests, the number of steps was increasing. For test 5 - 10 steps and test 6 - 1000 steps.
On the next page there are plots of calculated solutions and errors.
Here I observed the same behaviour as in previous tests.

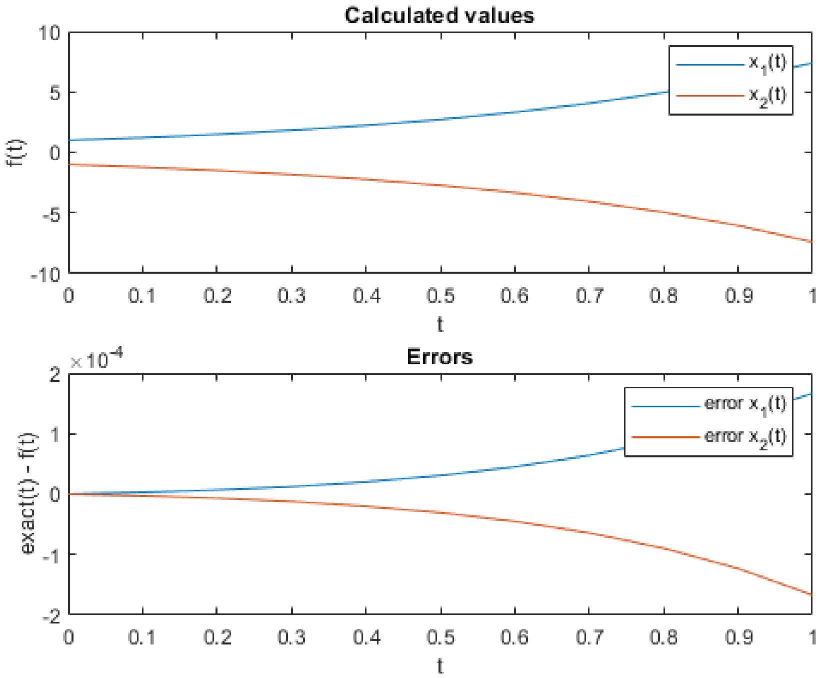

Figure 5: Test 5

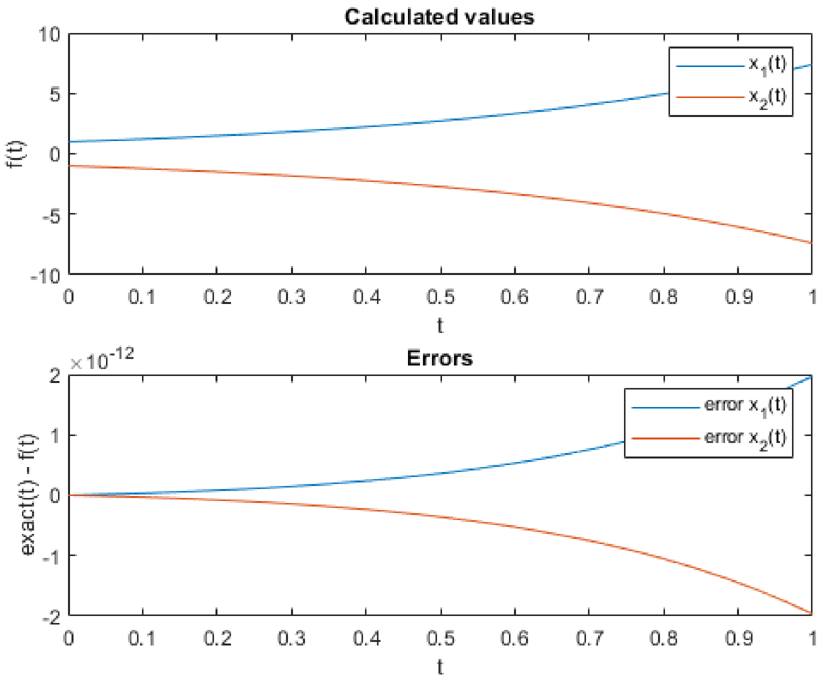

Figure 6: Test 6

Tests 7-10

For tests 7-10 I used following system of differential equations:

$$\begin{cases} x_1'(t) = -80.6x_1 + 119.4x_2 \\ x_2'(t) = 79.6x_1 - 120.4x_2 \end{cases}$$

with initial condition:

$$\begin{cases} x_1(0) = 3c_1 - c_2 \\ x_2(0) = 2c_1 + c_2 \end{cases}$$

and exact solution:

$$\begin{cases} x_1(t) = c_1 * 3e^{-t} - c_2e^{-200t} \\ x_2(t) = c_1 * 2e^{-t} + c_2e^{-200t} \end{cases}$$

This system of ODE is stiff, if $c_2 = 0$, then solution should be smooth, without significant error. However if $c_2 \neq 0$ it should produce significant error. For test 7, $c_1 = 5, c_2 = 0$, test 8 $c_1 = 5, c_2 = 1$, test 9 $c_1 = 1, c_2 = 0$ and test 10 $c_1 = 5, c_2 = 0$.
Plots of these tests are on following pages.
It can be seen, that results are as expected, meaning that Runge-Kutta-Gill method is not suitable for every type of ODE.

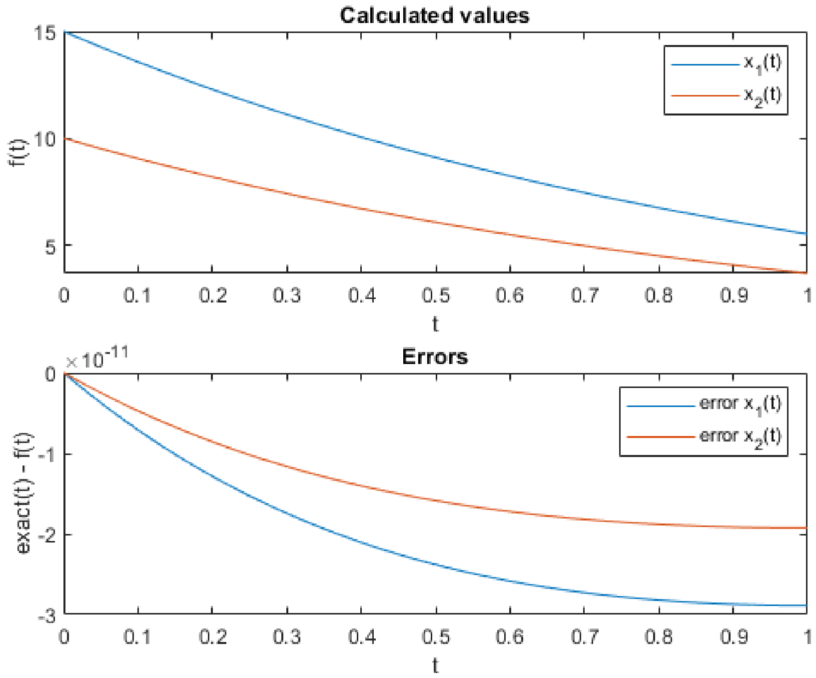

Figure 7: Test 7

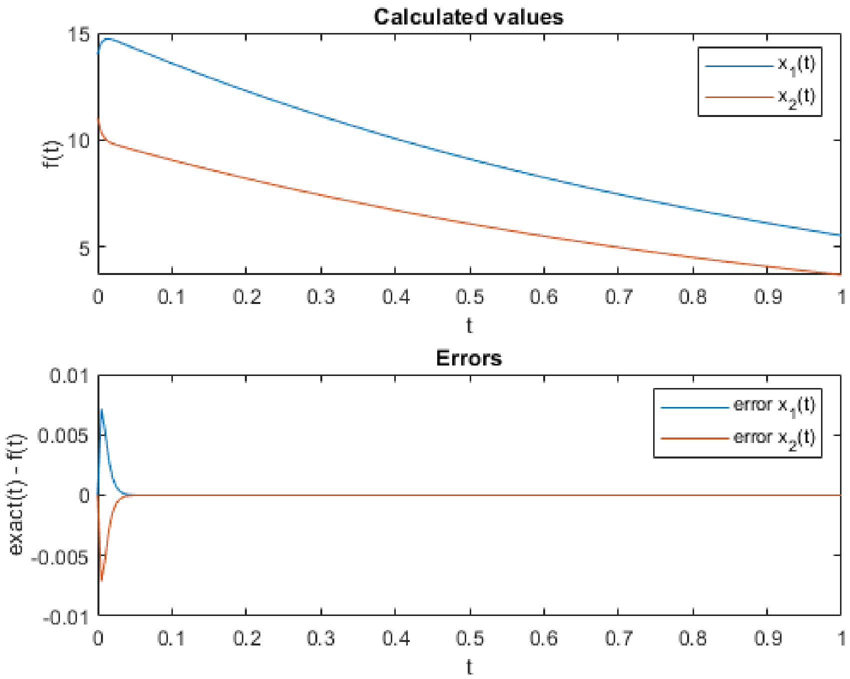

Figure 8: Test 8

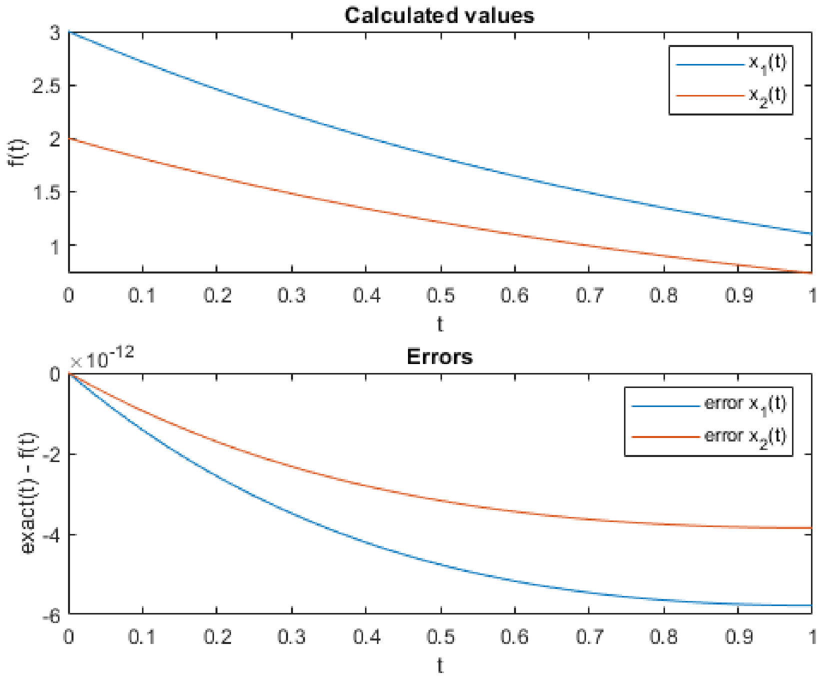

Figure 9: Test 9

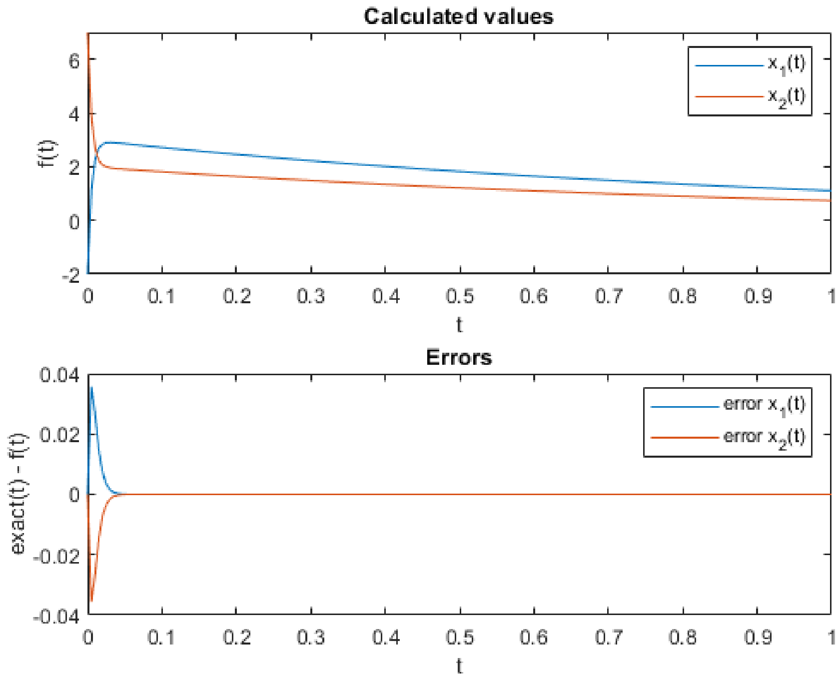

Figure 10: Test 10