TTK4130 - Modeling and Simulation
Assignement 7
Ingebrigt Stamnes Reinsborg

1a) Dynamic model:

$$\dot{x} = f(x, u, t)$$

Assume:

$$u = u_k \quad \text{for} \quad t \in [t_k, t_{k+1}]$$

ERK2 based on two evaluations of f, the first on $[x(t_k), u_k]$ can be written as:

$$K_1 = f(x(t_k), u(t_k), t_k)$$
$$K_2 = f(x(t_k) + a \cdot \Delta t \cdot K_1, \; u(t_k + c\Delta t), t_k + c\Delta t)$$

$$x_{k+1} = x(t_k) + \Delta t \sum_{i=1}^{2} b_i \cdot k_i$$

Butcher tableau

$$\begin{array}{c|cc} 0 & & \\ c & a & \\ \hline & b_1 & b_2 \end{array}$$

ERZ numerical error:

$$e_k = x_{k+1} - x(t_{k+1}|k)$$

$x(t|k)$ actual traj. of ODE with init. cond. $x(t_k|k) = x_k$.

I am to provide cond. on a, b_1, b_2 and c such that e_k of the method is of order 3.

$$x(t_{k+1}) = x(t_k) + \Delta t \cdot f(x(t_k), u_k)$$
$$+ \frac{\Delta t^2}{2} \cdot \dot{f}(x(t_k), u_k) + O(\Delta t^3)$$

(2c) $X_{k+1} = X(t_k) + \Delta t\,(b_1 K_1 + b_2 K_2)$

(3) $\Downarrow$

$$X_{k+1} - X(t_{k+1}) = \Delta t\, b_1 K_1 + \Delta t\, b_2 K_2 - \Delta t\, K_1 - \frac{\Delta t^2}{2} \underbrace{\dot f(x(t_k), u_k)}_{(\approx \dot f \text{ from now})} + O(\Delta t^3)$$

$$= e_k$$

We do a Taylor exp. on K_2

$$K_2 = K_1 + a\Delta t\, \dot f(X(t_k), u_k) + O(\Delta t^2)$$

$\Downarrow$

$$e_k = \Delta t\, b_1 K_1 + \Delta t\, b_2 K_1 + \Delta t^2 b_2 a \dot f + \Delta t\, b_2\, O(\Delta t^2) - \Delta t\, K_1 - \frac{\Delta t^2}{2} \dot f - O(\Delta t^3)$$

$$= \Delta t\,(b_1 + b_2 - 1) K_1 + \Delta t^2 (b_2 a - \tfrac{1}{2}) + O(\Delta t^3)$$

$$\Rightarrow \underline{\underline{b_1 + b_2 = 1 \;,\; b_2 a = \tfrac{1}{2} \;,\; c \in [0,1]}}$$

b) $e_K = O(\Delta t^3)$

$$\Downarrow$$

$$\|X_N - X(T)\| = O(\Delta t^2)$$

(If $\|X_{K+1} - X(t_{K+1})\|$, for each ~~step~~ step has an error $O(\Delta t^3)$ and integration to T, we get $\frac{T}{\Delta t} = N$ steps in total.

After these, the error is:

$$\frac{T}{\Delta t} \Delta t^3 = T \Delta t^2 = O(\Delta t^2)$$

c) Modus operandi for surviving third year spring-semester for MTTK, dictates that:

(optional) := "Do not"

Some day in the future, I'll have the time to enjoy what I learn.

2a) Butcher Tableau:

RK1:

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

RK2:

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

RK3:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & & \frac{1}{2} & & \\ 1 & & & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Test system:

$$\dot{x} = \lambda x$$

we'll use:

$$x_{k+1} = x_k + \Delta t \sum_i^s b_i K_i$$

$$K_s = f(x_k + \Delta t \sum_i^s a_{si} K_i \,,\, u(t_k + c_s \Delta t)$$

The code was tested for

$\lambda = -2$, $t \in [0,2]$, $\Delta t = 0.4$, $X(0) = 1$

See three first figures for RK1, RK2 and RK4

b) I got better results for lower Δt's

Very low Δt's (like 0.1) gave very good results for all methods (see the 4. 5. and 6. fig)

Actual order of methods as a function of Δt:

RK	Order	$\frac{1}{2}\Delta t$-error
1	1	$\frac{1}{2}$
2	2	$\frac{1}{4}$
4	4	$\frac{1}{16}$

c) At $\lambda < -5$ RK1 and 2 is unstable

At $\lambda < -6.97$ RK4 is unstable

(I just twiddled with the code for a bit until they became unstable.)

3a) $\dot{x} = y$

$\dot{y} = u(1-x^2)y - x$

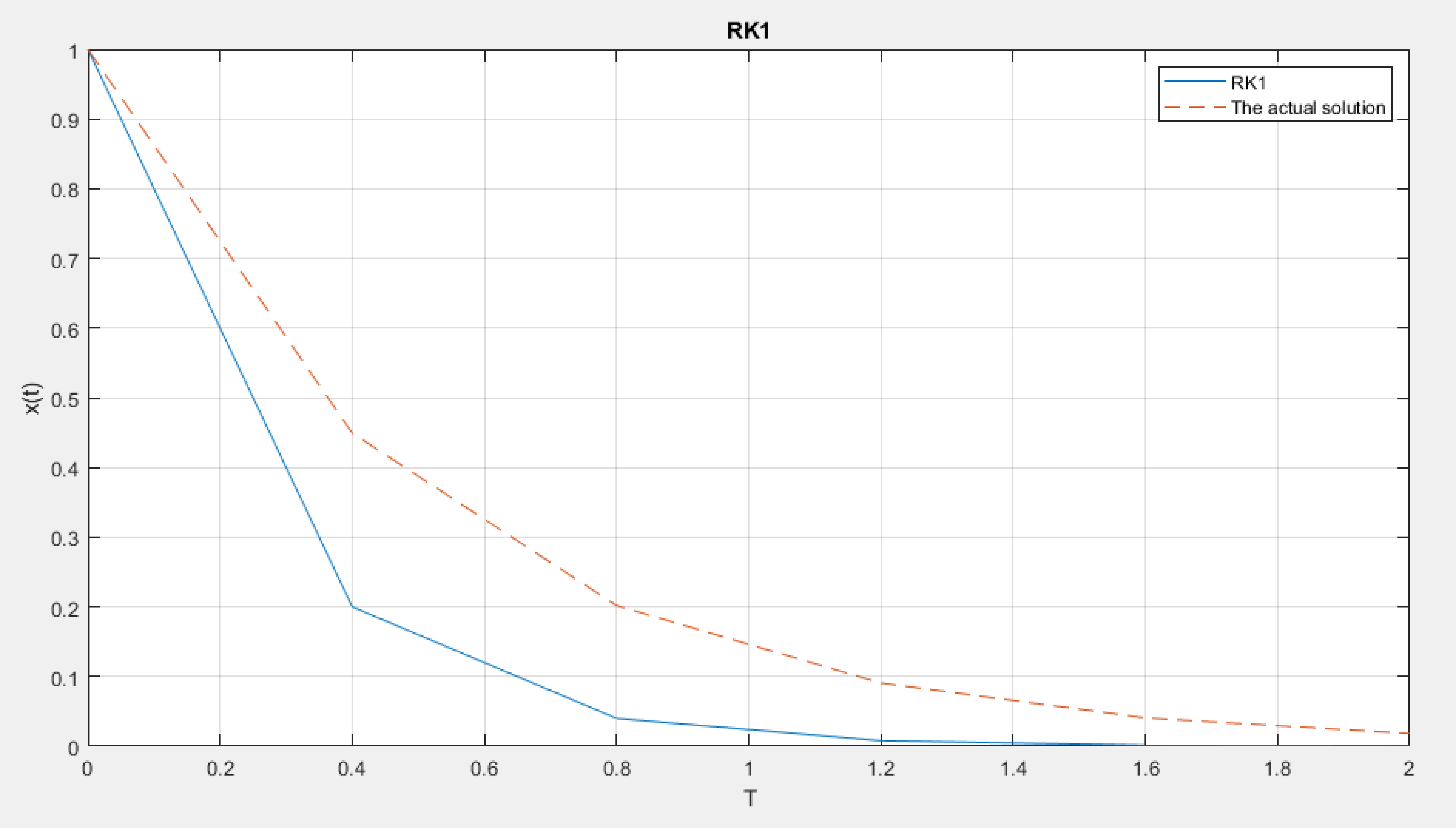
$u = 5$
$x(0) = 2$
$y(0) = 0$

With ODE45 it seems both x and y are marginally stable

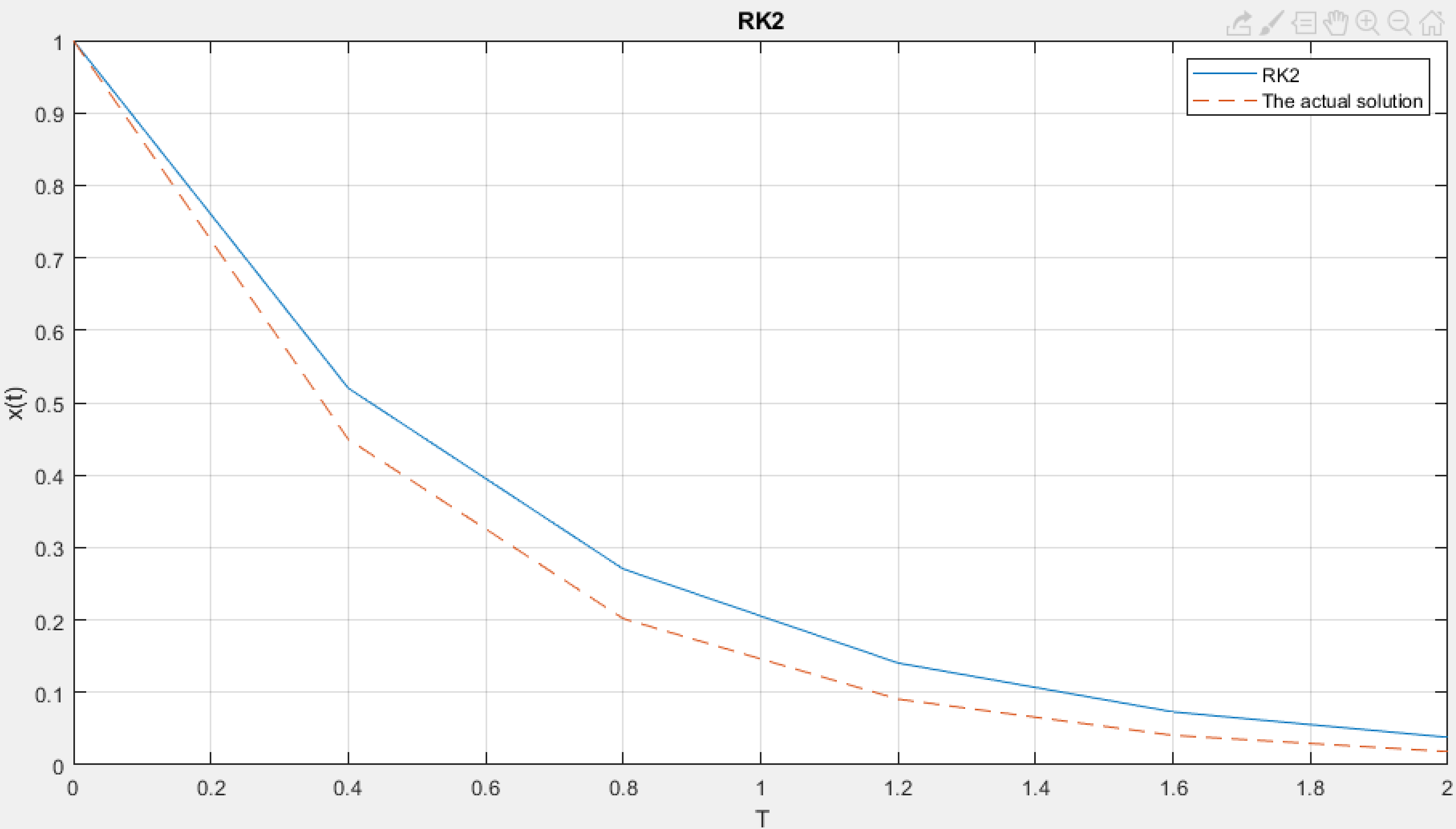
b) Increasing beyond $\Delta t = 16$ doesn't work.

Seems lower values work the best

$\Delta t < 0.12$ gives that RK4 and ODE45 gives somewhat equal plots.



RK1
RK1
The actual solution
x(t)
T
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2



RK2
RK2
The actual solution
x(t)
T
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

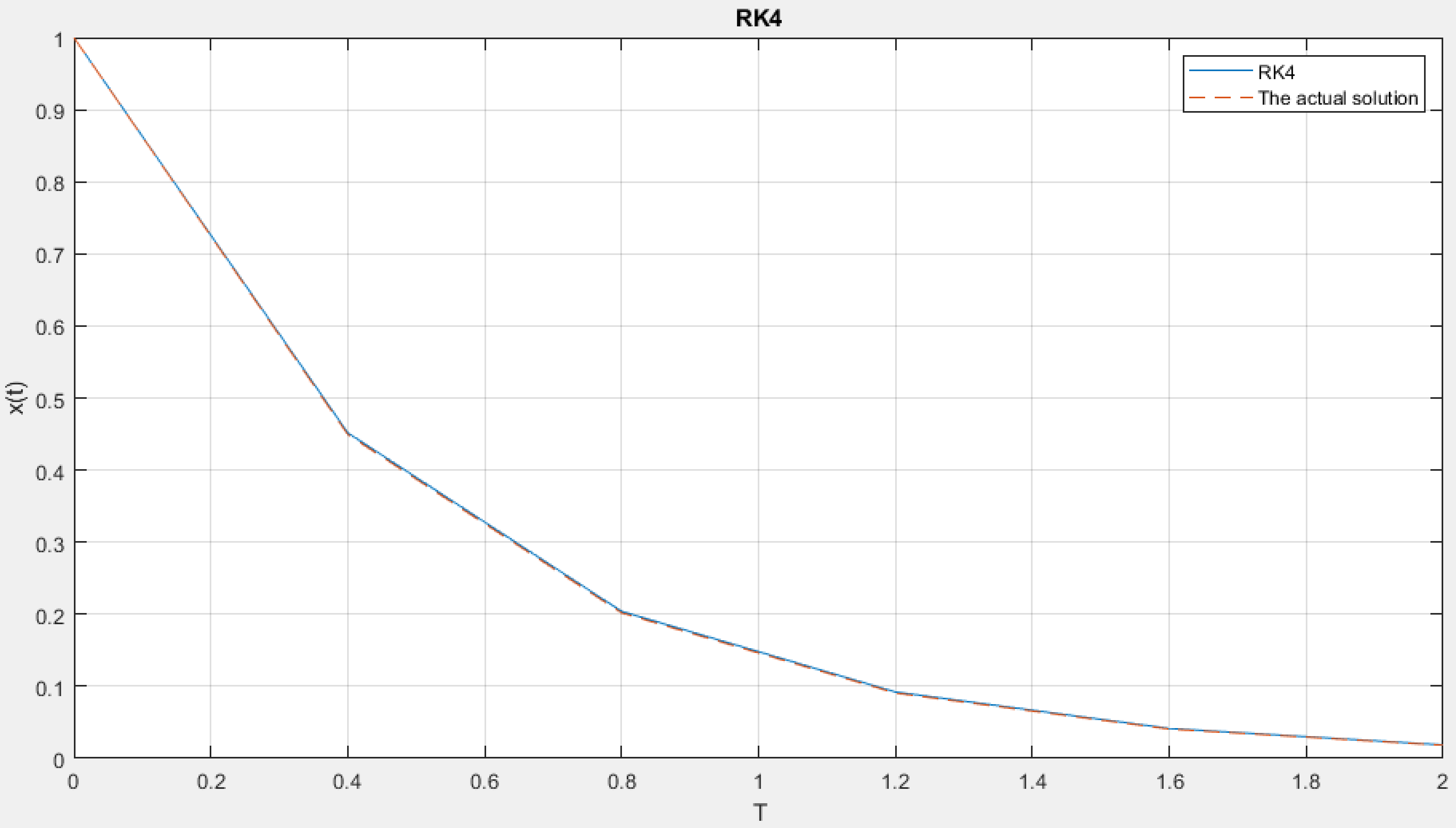

RK4
RK4
The actual solution
x(t)
T
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.2
0.4
0.6
0.8
1.2
1.4
1.6
1.8
2

Figure 2

File Edit View Insert Tools Desktop Window Help

$dt = 0,1$

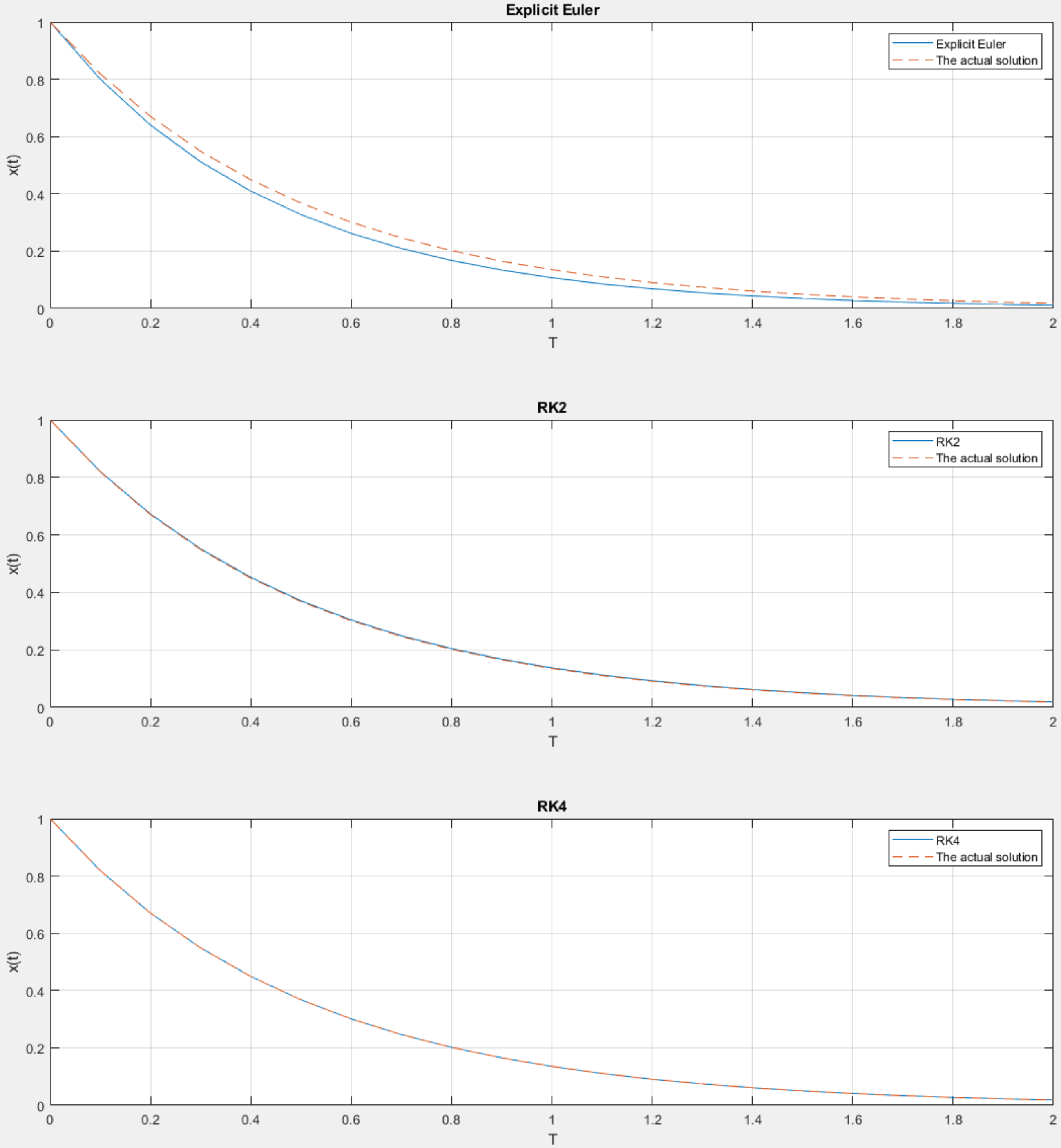


Figure 2

File Edit View Insert Tools Desktop Window Help

$dt = 0.8$

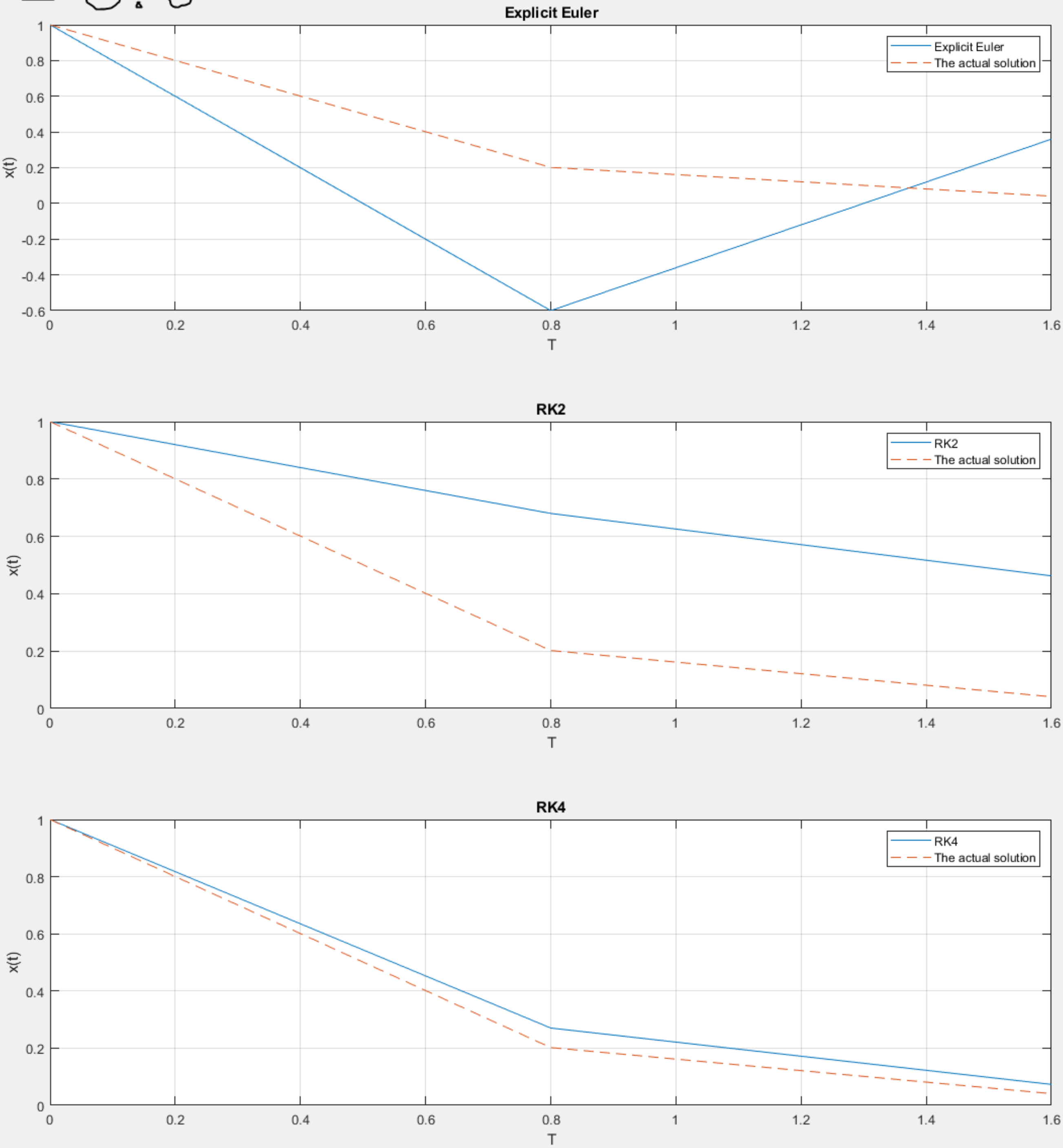
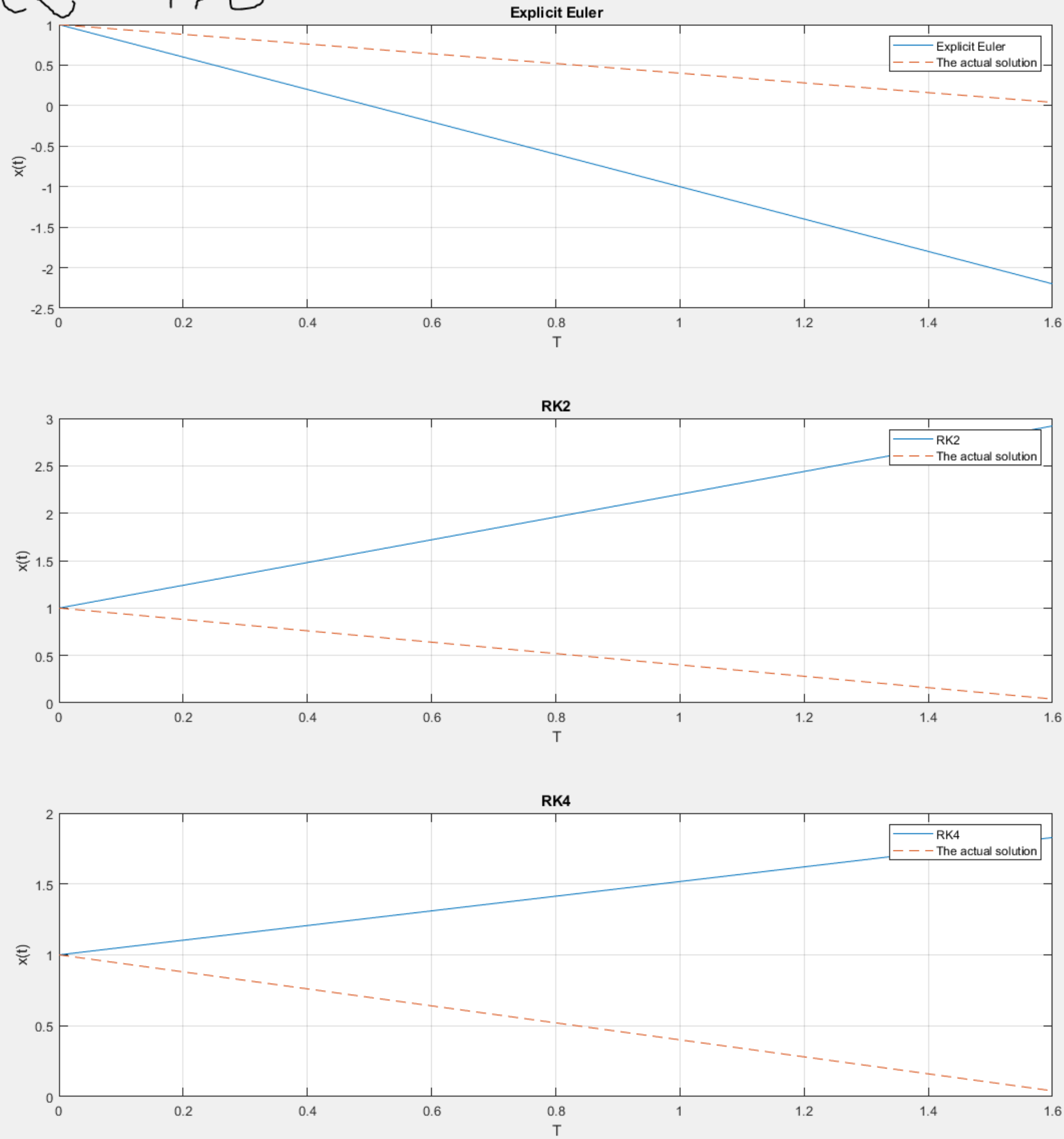


Figure 2

File Edit View Insert Tools Desktop Window Help

dt = 1.6



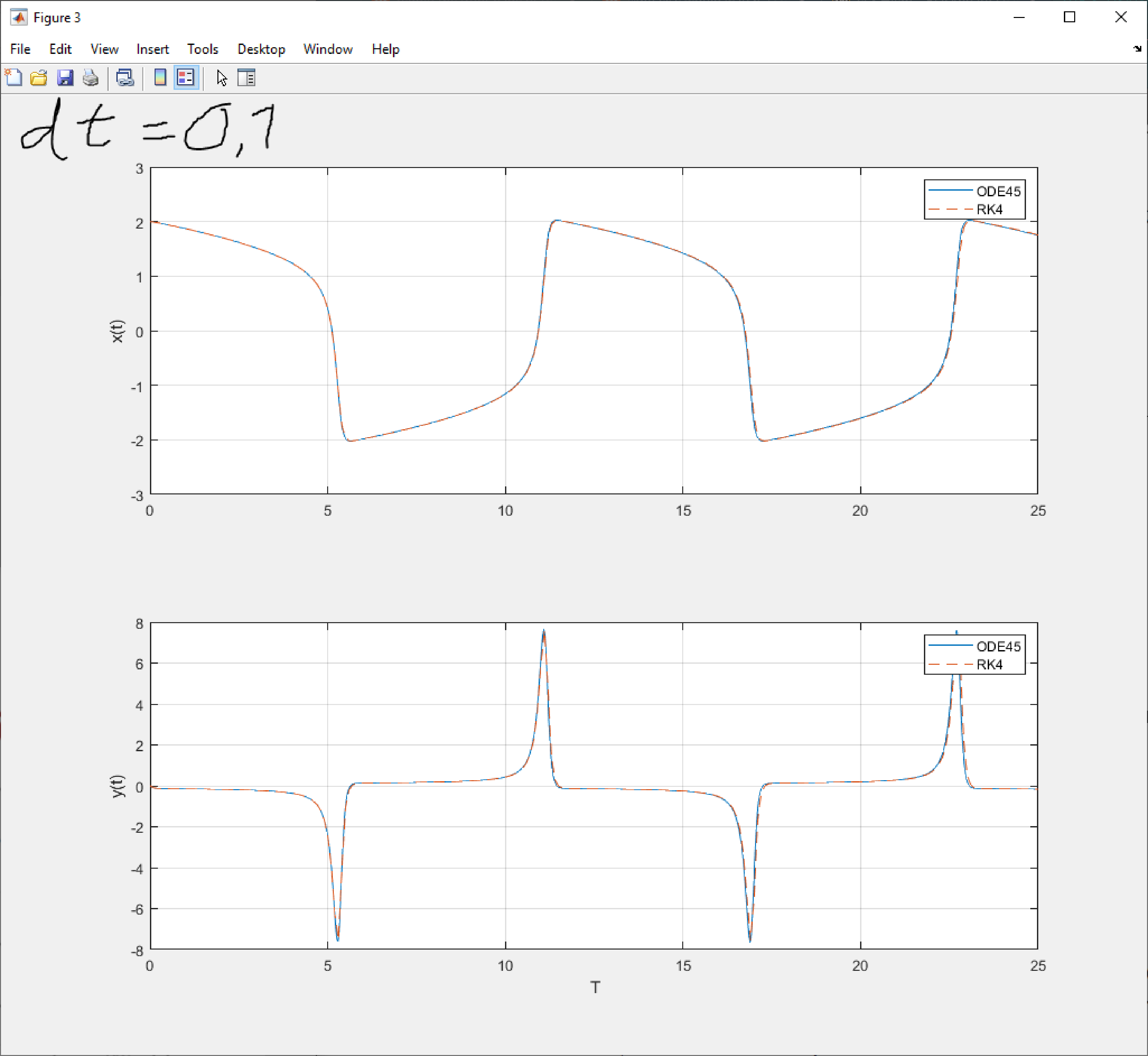
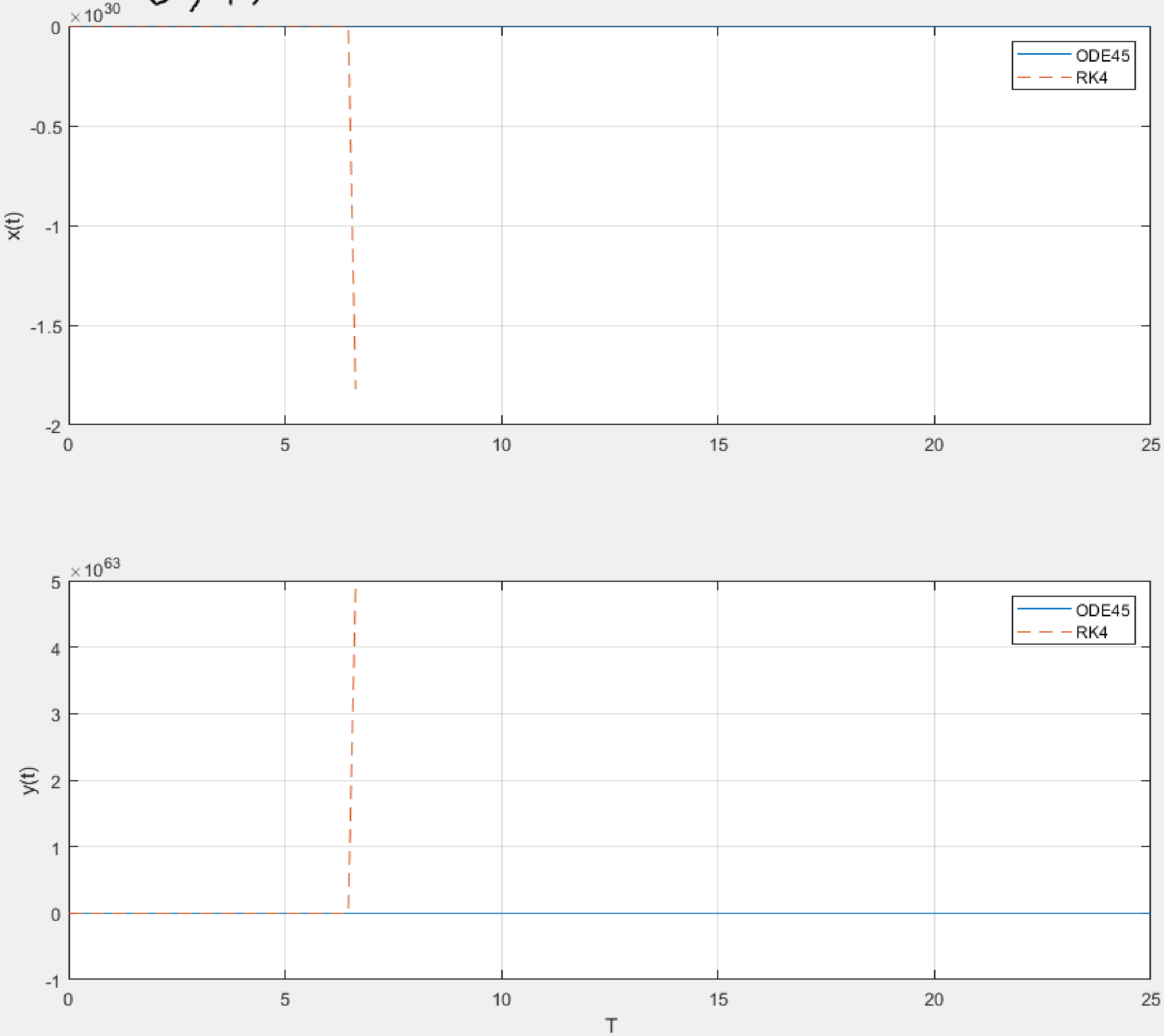


Figure 3
File Edit View Insert Tools Desktop Window Help
dt = 0,1
ODE45
RK4
x(t)
ODE45
RK4
y(t)
T

Figure 3

File Edit View Insert Tools Desktop Window Help

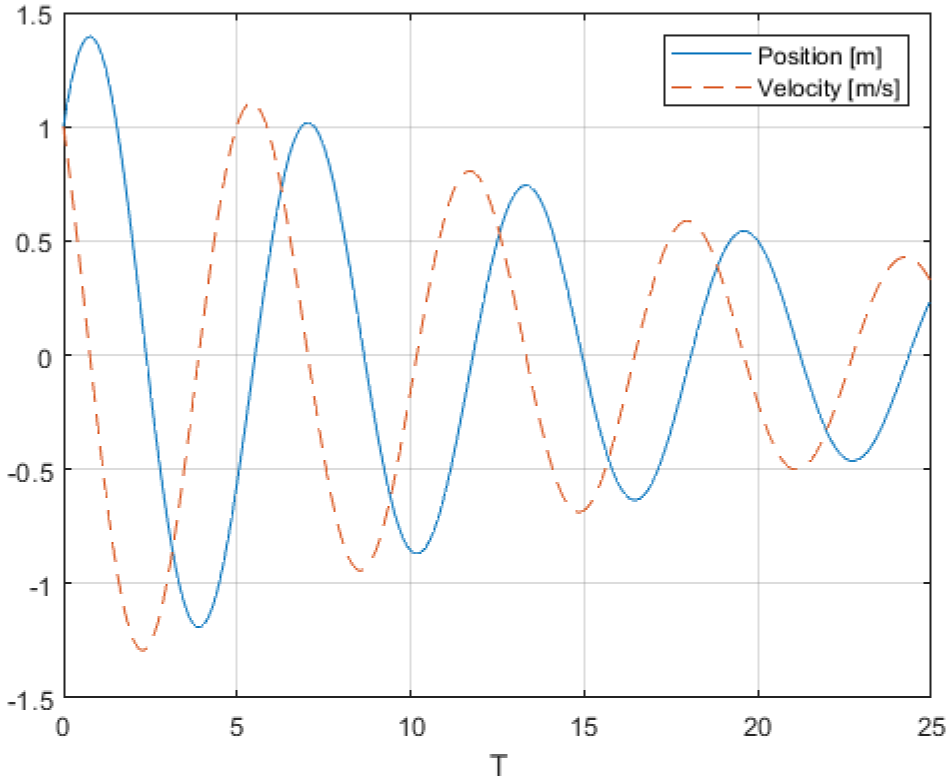
dt = 0,17

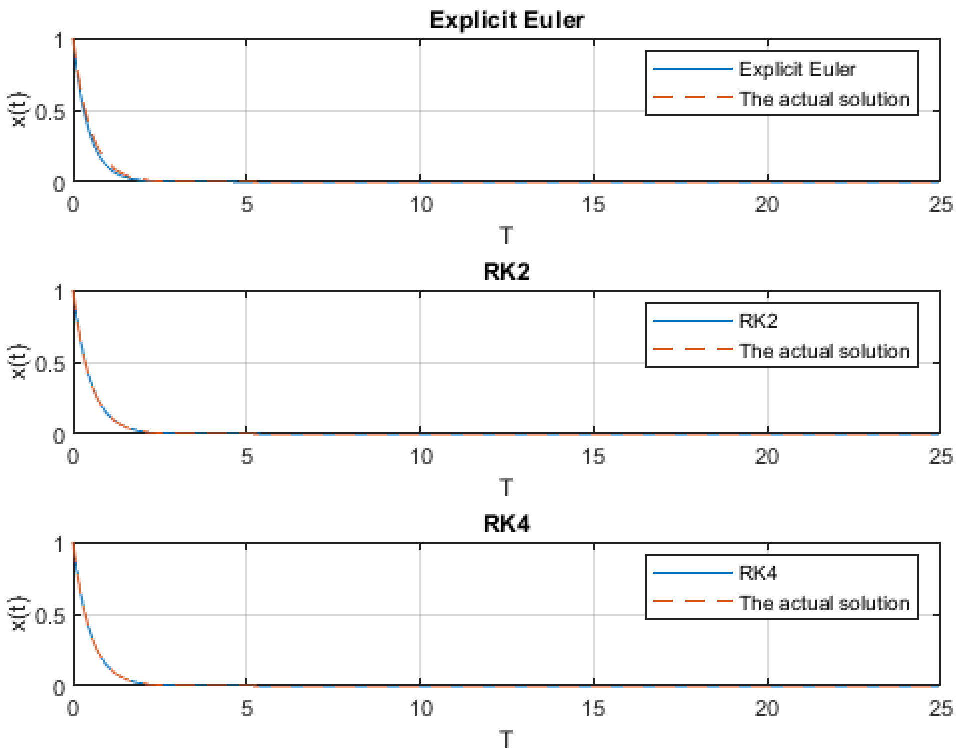


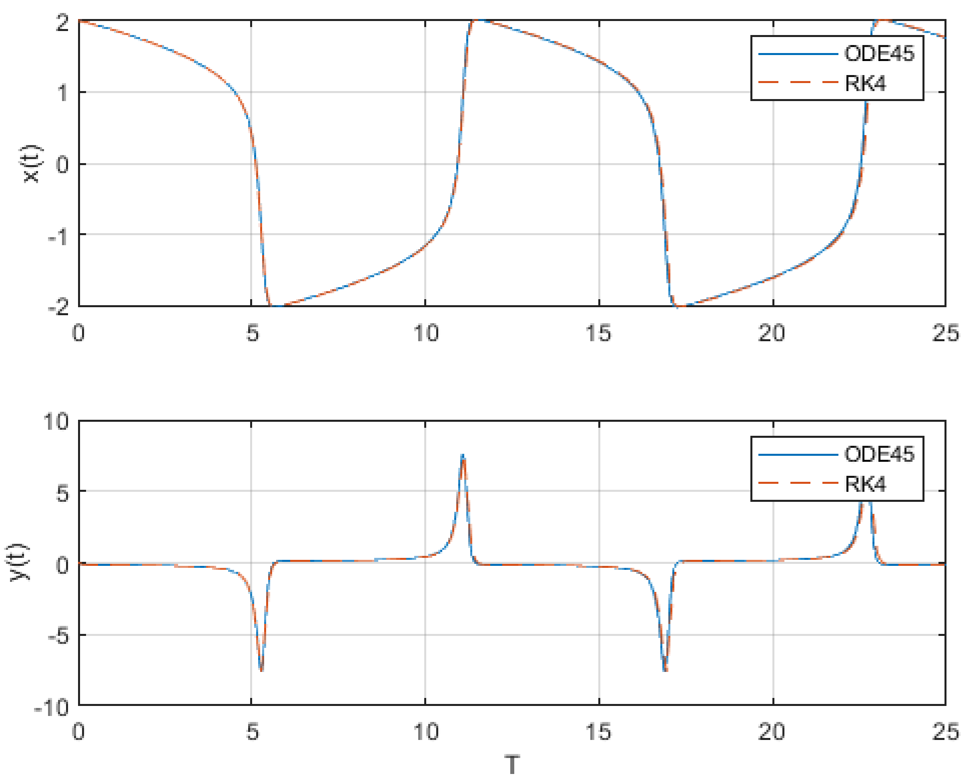
```
clear
clc
% Mass-damper-spring parameters
m = 1;
d = 0.1;
k = 1;
A = [ 0      1;
    -k/m -d/m];
% Mass-damper-spring vector field
fMassDamperSpring = @(t,x) A*x;

% Explicit Euler
A1 = 0;
c1 = 0;
b1 = 1;
RK1 = struct('A',A1,'b',b1,'c',c1);

% RK2
A2 = [0  0;
     1/2 0];
c2 = [0;
     1/2];
b2 = [0;
      1];
RK2 = struct('A',A2,'b',b2,'c',c2);

% RK4
A4 =  [0 0 0 0;
     1/2 0 0 0;
     0 1/2 0 0;
      0 0 1 0];
c4 = [0;
    1/2;
    1/2;
    1];
b4 = [1/6;
    1/3;
    1/3;
    1/6];
RK4 = struct('A',A4,'b',b4,'c',c4);


% Task 1-2 parameters
lambda = -2;
dT = 0.11;
T = 0:dT:25;
x0 = 1;
func = @(t,x) lambda*x;
actual_solution = exp(lambda*T);

% Simulate
X1 = ERKTemplate(RK1,func,T,dT,x0);
```

```
X2 = ERKTemplate(RK2,func,T,dT,x0);
X4 = ERKTemplate(RK4,func,T,dT,x0);
X_mass = ERKTemplate(RK2,fMassDamperSpring,T,dT,[1;1]);

% Task 2 plots

%mass damper spring
figure(1)
plot(T,X_mass(1,:),T,X_mass(2,:), '--');
legend('Position [m]','Velocity [m/s]');
xlabel('T')
grid on

%RK1,2,4
figure(2)
subplot(3,1,1)
plot(T,X1,T,actual_solution, '--');
legend('Explicit Euler','The actual solution');
ylabel('x(t)');
xlabel('T');
title('Explicit Euler');
grid on

subplot(3,1,2)
plot(T,X2,T,actual_solution, '--');
legend('RK2','The actual solution');
ylabel('x(t)');
xlabel('T');
title('RK2');
grid on

subplot(3,1,3)
plot(T,X4,T,actual_solution, '--');
legend('RK4','The actual solution');
ylabel('x(t)');
xlabel('T');
title('RK4');
grid on

% Task 3 vanderpol
u = 5;
state0 = [2;
          0];
t_final = 25;

[time,statetraj] = ode45(@(t,x)vanderpol(t, x, u),[0 t_final],
 state0);

vanderpol_func = @(t,x) vanderpol(t, x, u);

x_vdp = ERKTemplate(RK4,vanderpol_func,T,dT,state0);

%Task 3 Plot
figure(3)
```

```
subplot(2,1,1)
plot(time,statetraj(:,1),T,x_vdp(1,:), '--');
ylabel('x(t)'); legend('ODE45', 'RK4');
grid on

subplot(2,1,2)
plot(time,statetraj(:,2),T,x_vdp(2,:), '--');
ylabel('y(t)'); xlabel('T'); legend('ODE45', 'RK4');
grid on
```




Explicit Euler
Explicit Euler
The actual solution
x(t)
T
RK2
RK2
The actual solution
x(t)
T
RK4
RK4
The actual solution
x(t)
T


ODE45
RK4
x(t)
ODE45
RK4
y(t)
T

Published with MATLAB® R2019a

```matlab
function x = ERKTemplate(ButcherArray, f, T, dT, x0)
    % Returns the iterations of an ERK method
    % ButcherArray: Struct with the ERK's Butcher array
    % f: Function handle
    %    Vector field of ODE, i.e., x_dot = f(t,x)
    % T: Vector of time points, 1 x Nt
    % x0: Initial state, Nx x 1
    % x: ERK iterations, Nx x Nt
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Define variables
    % Allocate space for iterations (x) and k1,k2,...,kNstage
    % It is recommended to allocate a matrix K for all kj, i.e.
    % K = [k1 k2 ... kNstage]

    A = ButcherArray.A;
    c = ButcherArray.c;
    b = ButcherArray.b;

    Nstage = size(c,1);
    Nt = size(T, 2);
    Nx = size(x0, 1);

    K = zeros(Nx, Nstage);
    x = zeros(Nx, Nt);


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    x(:,1) = x0;
    % Loop over time points
    for nt=2:Nt
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Update variables
        x_k = x(:,nt-1);
        K(:,1) = f(T(nt), x_k);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Loop that calculates k1,k2,...,kNstage
        for nstage=2:Nstage
            ksum = 0;
            for i=1:nstage-1
                ksum = ksum + A(nstage,i)*K(:,i);
            end
            K(:,nstage) = f(T(nt), x_k+dT*ksum);
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Calculate and save next iteration value x_t
        xsum = 0;
        for m=1:Nstage
           xsum = xsum + b(m)*K(:,m);
        end
        x(:,nt) = x_k + dT*xsum;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end
```

```
end
```

```
Not enough input arguments.

Error in ERKTemplate (line 15)
    A = ButcherArray.A;
```

Published with MATLAB® R2019a

```
function [state_dot] = vanderpol( t, state, input )

%states and input
x = state(1);
y = state(2);
u = input;

%equations
x_dot = y;
y_dot = u*(1-x^2)*y-x;

state_dot = [x_dot; y_dot];
end
```

```
Not enough input arguments.

Error in vanderpol (line 4)
x = state(1);
```

Published with MATLAB® R2019a