Matlab Code

Note: • Question's and answers to problems are highlighted in code yellow/green.

• Graphs are shown on the last two pages.

Problems highlighted blue.

```
clear all;
close all;
clc;
format long;
name = 'Kasey Haman';
id = 'A16978114';
%%Problem 1
%Initialize for Euler-----------------------------------------------------------
f = @(t, y) exp(-y).*sin(t+2.*pi.*y);
T = 2; y(1) = 1;
%loop for number of steps and step size
n(1) = 4;
h(1) = T/n;
for l = 1:10
n(l+1) = n(l)*2;
h(l+1) = T/n(l+1);
end
%Solve Euler for each step size
for n = 1:11
i = 0;
y = 1;
for t = 0:h(n):2
i = i + 1;
y(i+1) = y(i) + h(n)*f(t, y(i));
end
Eulertvalues{n} = 0:h(n):2;
Euleryvalues{n} = y(1:end-1);
Eulerysoln(n) = y(end-1);
end
%loop for number of steps for graph.
figure(1), hold on;
for n = 1:11
str(n) = T./h(n);
legend_string{n} = sprintf('Euler with %d steps',str(n));
cs = 'bgrcmykbrgcm';
plot(Eulertvalues{n},Euleryvalues{n}, cs(n),'LineWidth',1);
end
legend(legend_string,'Location','Northwest');
xlabel('Time T'); ylabel('yk Approximation');
title('Euler Approximation');
box on; grid on;
set(gca,'FontSize',10)
%Initialize for RK4-----------------------------------------------------
f = @(t, y) exp(-y).*sin(t+2.*pi.*y);
T = 2; y(1) = 1;
%loop for number of steps and step size
n(1) = 4;
h(1) = T/n;
for l = 1:10
```

```
n(l+1) = n(l)*2;
h(l+1) = T/n(l+1);
end
%Solve RK4 for each step size
for n = 1:11
i = 0;
y = 1;
for t = 0:h(n):2
i = i + 1;
k1 = h(n)*f(t, y(i));
k2 = h(n)*f(t+h(n)/2, y(i)+k1/2);
k3 = h(n)*f(t+h(n)/2, y(i)+k2/2);
k4 = h(n)*f(t+h(n), y(i)+k3);
y(i+1) = y(i) + 1/6*(k1+2*(k2+k3)+k4);
end
RK4tvalues{n} = 0:h(n):2;
RK4yvalues{n} = y(1:end-1);
RK4ysoln(n) = y(end-1);
end
%loop for number of steps for graph.
figure(2), hold on;
for n = 1:11
str(n) = T./h(n);
legend_string{n} = sprintf('RK4 with %d steps',str(n));
cs = 'bgrcmykbrgcm';
plot(RK4tvalues{n},RK4yvalues{n}, cs(n),'LineWidth',1);
end
legend(legend_string,'Location','Northwest');
xlabel('Time T'); ylabel('yk Approximation');
title('RK4 Approximation');
box on; grid on;
set(gca,'FontSize',10)
%Initialize for RK2------------------------------------------------------
f = @(t, y) exp(-y).*sin(t+2.*pi.*y);
B1 = 1/2; B2 = 1/2; mew = 1; alpha = 1; T = 2; y(1) = 1;
%loop for number of steps and step size
n(1) = 4;
h(1) = T/n;
for l = 1:10
n(l+1) = n(l)*2;
h(l+1) = T/n(l+1);
end
%Solve RK2 for each step size
for n = 1:11
i = 0;
y = 1;
for t = 0:h(n):2
i = i + 1;
k1 = h(n)*f(t, y(i));
```

```
k2 = h(n)*f(t+mew*h(n), y(i)+alpha*k1);
y(i+1) = y(i) + B1*k1 + B2*k2;
end
RK2tvalues{n} = 0:h(n):2;
RK2yvalues{n} = y(1:end-1);
RK2ysoln(n) = y(end-1);
end
%loop for number of steps for graph.
figure(3), hold on;
for n = 1:11
str(n) = T./h(n);
legend_string{n} = sprintf('RK2 with %d steps',str(n));
cs = 'bgrcmykbrgcm';
plot(RK2tvalues{n},RK2yvalues{n}, cs(n),'LineWidth',1);
end
legend(legend_string,'Location','Northwest');
xlabel('Time T'); ylabel('yk Approximation');
title('RK2 Approximation');
box on; grid on;
set(gca,'FontSize',10)
%Initialize for Custom RK2------------------------------------------------
f = @(t, y) exp(-y).*sin(t+2.*pi.*y);
B1 = 3/4; B2 = 1/4; mew = 2; alpha = 2; T = 2; y(1) = 1;
%loop for number of steps and step size
n(1) = 4;
h(1) = T/n;
for l = 1:10
n(l+1) = n(l)*2;
h(l+1) = T/n(l+1);
end
%Solve RK2 Custom for each step size
for n = 1:11
i = 0;
y = 1;
for t = 0:h(n):2
i = i + 1;
k1 = h(n)*f(t, y(i));
k2 = h(n)*f(t+mew*h(n), y(i)+alpha*k1);
y(i+1) = y(i) + B1*k1 + B2*k2;
end
RKCtvalues{n} = 0:h(n):2;
RKCyvalues{n} = y(1:end-1);
RKCysoln(n) = y(end-1);
end
%loop for number of steps for graph.
figure(4), hold on;
for n = 1:11
str(n) = T./h(n);
legend_string{n} = sprintf('RK2 with %d steps',str(n));
```

```
cs = 'bgrcmykbrgcm';
plot(RKCtvalues{n},RKCyvalues{n}, cs(n),'LineWidth',1);
end
legend(legend_string,'Location','Northwest');
xlabel('Time T'); ylabel('yk Approximation');
title('Custom RK2 Approximation');
box on; grid on;
set(gca,'FontSize',10)
%Solve Runge Kuta 4 for "true" solution------------------------------------
f = @(t, y) exp(-y).*sin(t+2.*pi.*y);
T = 2; y(1) = 1; n = 8192; h = T/n;
i = 0;
for t = 0:h:2
i = i + 1;
k1 = h*f(t, y(i));
k2 = h*f(t+h/2, y(i)+k1/2);
k3 = h*f(t+h/2, y(i)+k2/2);
k4 = h*f(t+h, y(i)+k3);
y(i+1) = y(i) + 1/6*(k1+2*(k2+k3)+k4);
end
treuyvalues = y(1:end-1);
treutvalues = 0:h:2;
trueyvalue = treuyvalues(end);
%Initialize for error plot-----------------------------------------------
for n = 1:11
ErrorEuler(n) = abs(trueyvalue - Eulerysoln(n));
ErrorRK2(n) = abs(trueyvalue - RK2ysoln(n));
ErrorRKC(n) = abs(trueyvalue - RKCysoln(n));
ErrorRK4(n) = abs(trueyvalue - RK4ysoln(n));
end
n(1) = 4;
h(1) = T/n;
for l = 1:10
n(l+1) = n(l)*2;
h(l+1) = T/n(l+1);
end
%Plot the log error
figure(5), hold on;
cs = 'krbgmckrbgm';
plot(log10(n),log10(ErrorEuler), '-k','LineWidth',1);
plot(log10(n),log10(ErrorRK2), 'r','LineWidth',1);
plot(log10(n),log10(ErrorRKC), 'b','LineWidth',1);
plot(log10(n),log10(ErrorRK4), 'c','LineWidth',1);
xlabel('Log10(n)'); ylabel('log10(Error y(T))');
title('Log Error y(T) versus log(n)');
legend('Euler', 'RK2', 'Custom RK2', 'RK4','Location','Southwest');
box on; grid on;
set(gca,'FontSize',10)
%Q: Comment on the slopes
```

```matlab
%Ans: We see that RK4 has the highest slope, the RK2 functions (which look
%nearly identical) have the second highest slope and our euler function has
%the lowest slope. The slope indicates how fast a function will converge to
%the correct solution, with a higher slope indicating faster convergence.
%Thus, RK4 is the fastest among this groupset.
%%Problem 2--------------------------------------------------------------
%Plot the log error in y(T) versus log f
%Note: Becuase Rk4 has 4 function evals per step, the amount of function
%evals is n*4. Similarly RK2 has 2 function evals per step (n.*2) and
% Euler's only has one per step (n).
figure(6), hold on;
cs = 'krbgmckrbgm';
plot(log10(n),log10(ErrorEuler), '-k','LineWidth',1);
plot(log10(n.*2),log10(ErrorRK2), 'r','LineWidth',1);
plot(log10(n.*2),log10(ErrorRKC), 'b','LineWidth',1);
plot(log10(n.*4),log10(ErrorRK4), 'c','LineWidth',1);
xlabel('Log10(Function Evals)'); ylabel('log10(Error y(T))');
title('Log Error y(T) versus Function Evaluations');
legend('Euler', 'RK2', 'Custom RK2', 'RK4','Location','Southwest');
box on; grid on;
set(gca,'FontSize',10)
%Q: How do the resulting plots change? Explain why they change the way they
%do.
%Ans: Certain plots are translated to the right. This means that there is a
%higher cost to get the same error. Note that because RK4 has more function
%evals per step, it has a higher translation than RK2. And because Euler
%has one function eval per step, it has no translation.
%%Problem 3--------------------------------------------------------------
clear all
%loop for number of steps and step size
T = 6;
n(1) = 8;
h(1) = T/n;
for l = 1:6
n(l+1) = n(l)*2;
h(l+1) = T/n(l+1);
end
%Initialize
f = @(z, t, y) exp(-1-sin(z))-sin(t+y).^2.*(1+z.^2).^(1/3);
y(1) = 4;
for n = 1:7
[yk_values{n},t_values{n}] = RK4(f, y, 0, h(n));
end
%Plot solution approximations
figure(7), hold on;
for n = 1:7
str(n) = T./h(n);
legend_string{n} = sprintf('RK4 with %d steps',str(n));
cs = 'bgrcmykbrgcm';
```

```
plot(t_values{n},yk_values{n}, cs(n),'LineWidth',1);
end
legend(legend_string,'Location','southwest');
xlabel('Time T'); ylabel('yk Approximation');
title('Problem 3 RK4 Approximation');
box on; grid on;
set(gca,'FontSize',10)
%Q: Explain why you selected the various step sizes. Would it make sense to
%have relation between the number of Runge Kutta steps and the number of
%steps in the fixed point component?
%A: I chose step sizes that varied up to 500 steps in order to see if the
%system would converge quickly with a low step size, using 500 steps as a true
%solution. It makes sense to ensure that the step sizes between fixed point
%and Runge Kutta are similar, as in real life we don't have infinite
%computing power. So we should practice with having equal calculation for
%both fixed point method and Runge Kutta in order to minimize
%error.
```

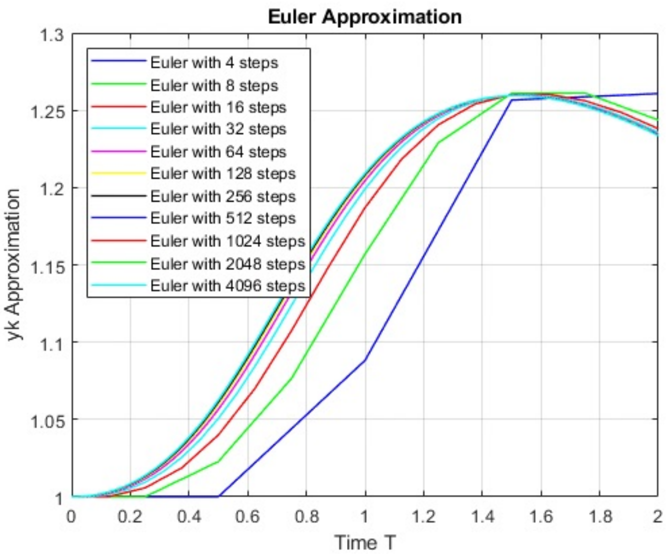
Function for problem 3:

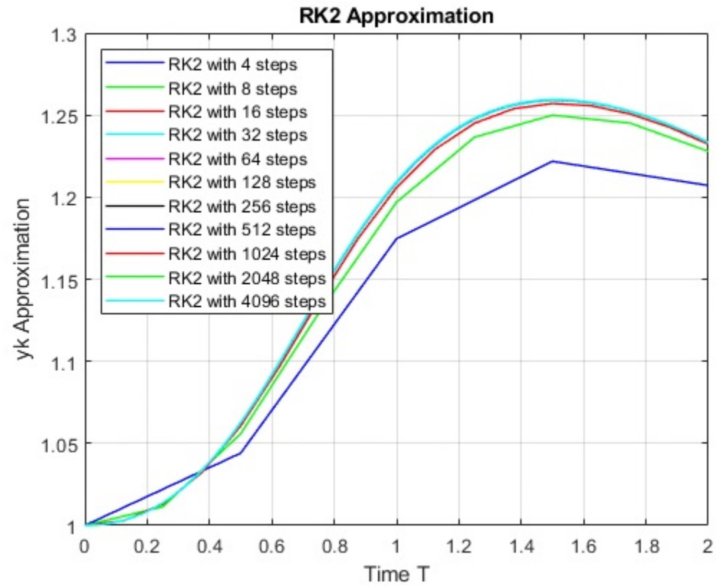
```
function [yk_values,t_values] = RK4(f, y, z, h)
%This is a Runge Kutta 4 function designed to find the root of a given
%numerical function.
%Note that this function includes the fixed point method used for solving
%the root of a function given 3 variables with 1 unknown.
%Run Runge Kuta 4 loop
i = 0; y(1) = y; z(1) = z;
for t = 0:h:6
i = i + 1;
k1 = h*f(z, t, y(i));
k2 = h*f(z, t+h/2, y(i)+k1/2);
k3 = h*f(z, t+h/2, y(i)+k2/2);
k4 = h*f(z, t+h, y(i)+k3);
y(i+1) = y(i) + 1/6*(k1+2*(k2+k3)+k4);
z = f(z, t, y(i));
end
yk_values = y(1:end-1);
t_values = 0:h:6;
end
```

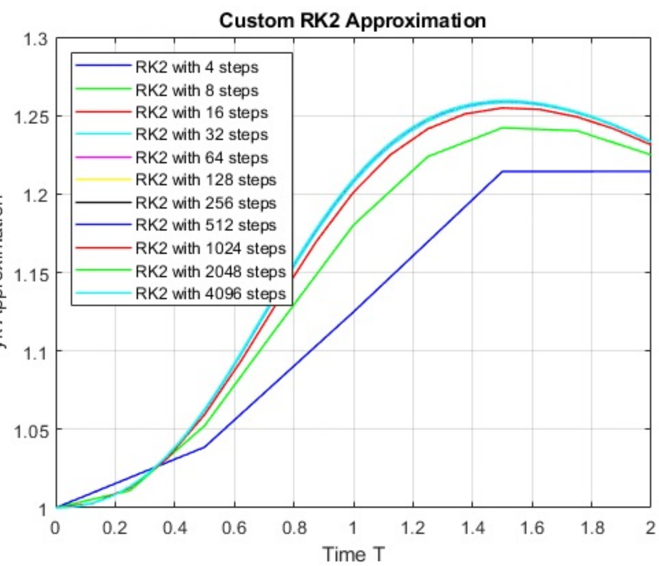
Graph's produced:

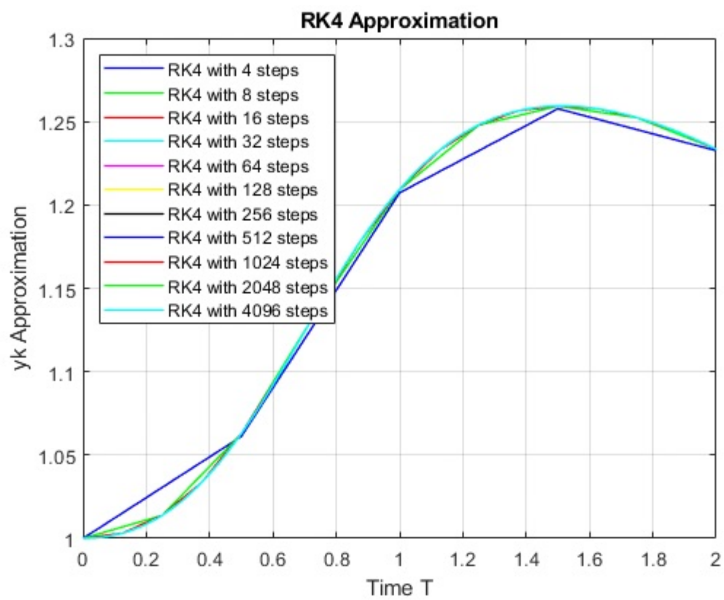
Note: Functions converge to approximately 1.233

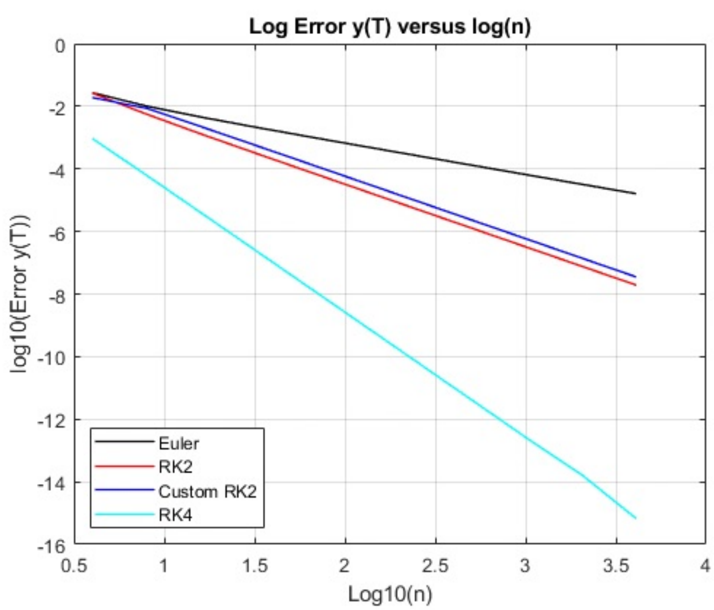
Problem 1)



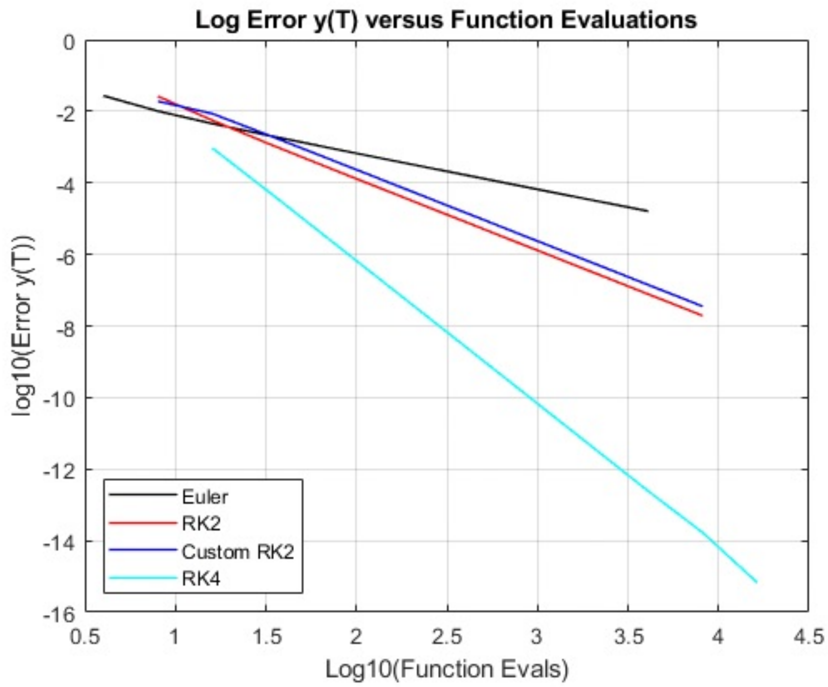








Problem 2)



Problem 3)

Note: Function converges to approximately 2.451