Table of Contents

.... 1
Workspace Prep 1
PART 1: Using RKF45 to Solve an ODE 1
PART 2: Lorenz Equation 2
Lorenz Function 3
Runge-Kutta-Fehlberg Function 3

```
%Roshan Jaiswal-Ferri
%Section - 03
%Aero 300 Lab 7 - Runge-Kutta-Fehlberg Methods: 5/17/24
```

Workspace Prep

```
format long      %Allows for more accurate decimals
close all;       %Clears all
clear all;       %Clears Workspace
clc;             %Clears Command Window
```

PART 1: Using RKF45 to Solve an ODE

```
%Given Parameter
t = [0, pi/3];
y0 = 1;
h = 0.01;
rtol = 1e-6;

%Given ODE
fun = @(t, y) (y - t - 1)^2 + 2;

%Solution to ODE (solution function)
solf = @(t) tan(t) + t + 1;

%Using rkf45
[t1, y1] = rkf45(fun, t, y0, h, rtol);

%Using ode45
options = odeset('RelTol', rtol, 'AbsTol', 1e-6);
[t2, y2] = ode45(fun, t, y0, options);

%rkf45 Error
for i = 1:length(t1)
    sol = solf(t1(i));
    rkfErr(i) = abs(y1(i) - sol); % Calculate the error for rkf45 at given
point
end

%ode45 Error
```

```
for i = 1:length(t2)
    sol = solf(t2(i));
    odeErr(i) = abs(y2(i) - sol); % Calculate the error for ode45 at given
point
end

%Plotting Results
figure('Name','Solution Plot')
plot(t1, y1, 'b');
hold on;
grid on;
plot(t2, y2, 'r--');
plot(t1, solf(t1), 'g:');
xlabel('Time');
ylabel('Calculated Solution');
legend('RKF45','ODE45','Actual Solution')
title('Results of RKF45, ODE45, and the Actual Solution of the ODE');

figure('Name','Error Comparison')
subplot(2,1,1)
plot(t1, rkfErr);
grid on;
xlabel('Time');
ylabel('Error');
title('Error of RKF45');

subplot(2,1,2)
plot(t2, odeErr, 'r');
grid on;
xlabel('Time');
ylabel('Error');
title('Error of ODE45');
```

PART 2: Lorenz Equation

```
%New Parameters
t = [0 50];
y0 = [1; 1; 1];
h = 0.01;
rTol = 1*10^-6;

%using rkf45
[t3, y3] = rkf45(@lorenz, t, y0, h, rTol);

%This is just to reorganize the data vector, long story short the output of
%rkf45 is one column that is super long that contains all 3 columns (x y z)
%data. In this super long column every-other-other starting from 1 is x,
%every-other-other starting from 2 is y, and every-other-other starting
%from 3 is z. This code takes all the data and reorganizes it into
%a new matrix that has 3 columns and assigns each data point to where its
%proper column coordinate (x y z)

totalPts = length(y3);
```

```
numRows = totalPts / 3;
y4 = zeros(numRows, 3);
for i = 1:numRows
    y4(i, 1) = y3((i-1)*3 + 1);
    y4(i, 2) = y3((i-1)*3 + 2);
    y4(i, 3) = y3((i-1)*3 + 3);
end

%Plotting Results
figure('Name','Lorenz Equation')
plot3(y4(:,1), y4(:,2), y4(:,3), 'r');
title('Lorenz Equation Solution using RKF45');
grid on;
xlabel('X');
ylabel('Y');
zlabel('Z');
```

Lorenz Function

```
function deriv = lorenz(t, y)
    %Parameters
    sig = 10;
    rho = 28;
    beta = 8/3;

    %EQs
    deriv = [sig * (y(2) - y(1)); y(1) * (rho - y(3)) - y(2);
        y(1) * y(2) - beta * y(3)];
end
```

Runge-Kutta-Fehlberg Function

```
function [t, y] = rkf45(fun, tspan, y0, h, rTol)
    f = fun;
    t0 = tspan(1);
    t2 = tspan(2);
    t = t0;
    y = y0;
    t1 = t;
    Wn1 = y;

    while t(end) < t2
        s1 = f(t1, Wn1);
        s2 = f(t1 + (1/4)*h, Wn1 + (1/4)*h*s1);
        s3 = f(t1 + (3/8)*h, Wn1 + (3/32)*h*s1 + (9/32)*h*s2);
        s4 = f(t1 + (12/13)*h, Wn1 + (1932/2197)*h*s1 - (7200/2197)*h*s2 +
(7296/2197)*h*s3);
        s5 = f(t1 + h, Wn1 + (439/216)*h*s1 - 8*h*s2 + (3680/513)*h*s3 -
(845/4104)*h*s4);
        s6 = f(t1 + (1/2)*h, Wn1 - (8/27)*h*s1 + 2*h*s2 - (3544/2565)*h*s3 +
(1859/4104)*h*s4 - (11/40)*h*s5);

        %4th and 5th order estimate
        WNext = Wn1 + h*((25/216)*s1 + (1408/2565)*s3 + (2197/4104)*s4 -
(1/5)*s5);
        ZNext = Wn1 + h*((16/135)*s1 + (6656/12825)*s3 + (28561/56430)*s4 -
(9/50)*s5 + (2/55)*s6);

        %Error
        err = norm(ZNext - WNext);

        %Error test
        if err / abs(WNext) < rTol
            t = [t; t1 + h];
            y = [y; WNext];
            Wn1 = ZNext;
            t1 = t1 + h;
        end

        %Changing step size
        h = 0.8 * ((rTol/err)^(.2)) * h;
    end
end
```

Published with MATLAB® R2023b