**Vaasa University**

**of**

**Applied Science**

**IITB3004-2 Basics of Mathematical Software**

**Project on**

**Runge-Kutta method for a DE of 1st degree of**

**y'(x)=-sin(x), when y(0)=2, x=0..4\*pi by using Matlab**

**Submitted to**

**Jussi Ojanen**

**Submitted by**

**Md Farukul Islam**

**ID: e1701764**

The first order Runge-Kutta method is the most basic [explicit method](https://en.wikipedia.org/wiki/Explicit_and_implicit_methods) for [numerical integration of ordinary differential equations](https://en.wikipedia.org/wiki/Numerical_ordinary_differential_equations) and is the simplest  [method](https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_method). The aim of this method is to finding methods that seemed to be optimal in terms of local truncation error and to finding built-in error estimators.

% Runge-Kutta method implementation

function [x,y] = rk(x0,xf,y0,n)

h = (xf - x0)/n; % step size h and sample size n

x = x0; % initial x

y = y0; % initial y

for i = 1:n

k1 = ODE(x0,y0); % left-hand slope

k2 = ODE(x0+h/2,y0+h\*k1/2); % 1st midpoint slope

k3 = ODE(x0+h/2,y0+h\*k2/2); % 2nd midpoint slope

k4 = ODE(x0+h,y0+h\*k3); % right-hand slope

k = (k1+2\*k2+2\*k3+k4)/6; % average slope

y0 = y0 + h\*k; % Runge-Kutta step

x0 = x0 + h; % new x

x = [x;x0]; % update x-column

y = [y;y0]; % update y-column

end

end

%ODE

function [dy] = ODE(x,y)

dy=-sin(x);

clf;

end

%Graphical representation of Runge-Kutta method for -sinx

[x,y] = rk(0,4\*pi,2,100)

plot(-sin(x))

title('Graphical representation of 1st order runga-kutta method for -sinx')

xlabel('x')

ylabel('y')



Fig 1: Graphical representation of Runge-Kutta method for -sin(x)

First, implemented the Runge-Kutta method by using first order ordinary differential equation method. For doing this first we need to define the initial value of the function variable and step size. Then calculate the several slopes and finally take the average value of the slopes. Then by using the slope value implement the Runge-Kutta method. In this case the first order ordinary differential equation function is -sin(x) and we use ordinary differential equation to solve the function. Finally, by using this function we implement the Runge-Kutta steps. The Fig 1 shows the graphical representation of Runge-Kutta method for the first order ordinary differential equation for the function of -sin(x) and in this case our initial value for x is 0 and final value is 4\*pi and for initial x the value of y is 2 as per our desired implementation range and calculate the other values for y using the initial value of x and this case we take the sample size is n = 100. The accuracy of implementation depends on the sample size so as much we take the sample size the calculation will reach the maximum accuracy level and it will decrease the error rate as well.

% absolute error and relative error calculation

f=@(x)cos(x)

x0=pi/4;

f\_exact=-sin(x0)

for i=1:21;

h(i)=10^(1-i);

f\_approx=(f(x0+h(i))-f(x0-h(i)))./(2\*h(i));

abserr(i)=abs(f\_approx-f\_exact);

relerr(i)=abs(f\_approx-f\_exact)/f\_exact;

end

figure(1)

subplot(211)

plot(h,abserr,'ro-');

xlabel('x')

ylabel('y')

title('absolute error calculation in normal scale')

subplot(212)

loglog(h,abserr,'ro-');

xlabel('x')

ylabel('y')

title('absolute error calculation in log scale')

figure(2)

subplot(211)

plot(h,relerr,'o--');

xlabel('x')

ylabel('y')

title('relative error calculation in normal scale')

subplot(212)

loglog(h,relerr,'o--');

xlabel('x')

ylabel('y')

title('relative error calculation in log scale')



Fig 2: Calculation of absolute error in normal scale and log scale



Fig 3: Calculation of relative error in normal scale and log scale

For the error calculation we use an approximate value with respect to our exact value for first order ordinary differential equation function -sin(x) and then calculate the absolute error and relative error. In this case we use the formula for absolute error calculation and relative error calculation.

abserr =abs(f\_approx-f\_exact)

relerr =abs(f\_approx-f\_exact)/f\_exact

From Fig 2 and Fig 3 we can see calculating our absolute error and relative error for first order ordinary differential equation for function -sin(x), we use the value of x is pi/4 as an approximate value and we calculate 21 value points for our error calculation. Finally, the figures show our error calculation for absolute error and relative error with a graphical representation in normal and log scale. It also shows that the error depends on the number of digit taken after the decimal point for the error calculation. For calculating error as much value, we can take after the decimal point as much the less error will show in our calculation.

Finally, we see that the main advantages of Runge-Kutta methods are that they are easy to implement, they are very stable and they are self-starting because unlike muti-step methods, we do not have to treat the first few steps taken by a single-step integration method as special cases. The primary disadvantages of Runge-Kutta methods are that they require significantly more computer time than multi-step methods of comparable accuracy and they do not easily yield good global estimates of the truncation error. However, for the straightforward dynamical systems under investigation in this course the advantage of the relative simplicity and ease of use of Runge-Kutta methods far outweighs the disadvantage of their relatively high computational cost.