

```

% MIT License
% Copyright (c) 2017 Samira C. Oliva
% Permission is hereby granted, free of charge, to any person obtaining a copy of this software and
% associated documentation files (the "Software"), to deal in the Software without restriction,
% including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense,
% and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so,
% subject to the following conditions: The above copyright notice and this permission notice shall be
% included in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT
% LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN
% NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
% WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
% SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
%
% Engineer: Samira C. Oliva
% Create Date: 05/13/2017
% Project Name: Numerical Methods to Approximate IVPs (for MATH143C)
% Description: Methods implemented are Euler (Taylor Order 1), Taylor Order 2, Runge-Kutta Order 4.

```

Eulers_method.m

```

% Created by Samira C. Oliva on 05/13/17
% Copyright (c) 2017 Samira C. Oliva. All rights reserved.
% Euler's Method to Solve IVP
function [t,w] = Eulers_method(a,b,N,alpha,fstring)
h = (b-a)/N; %step size
f = inline(fstring, 't', 'y'); %create function of diff. eq
t = linspace(a,b,N+1); %vector of N + 1 mesh points
w = zeros(size(t)); %array of approximations of size same as t
w(1) = alpha; %initial condition
for i=1:N %loop to compute the N remaining approximation
    w(i+1) = w(i) + h*f(t(i),w(i)); %difference equation to compute ith+1 approximations
end
end

```

TaylorOrderTwo.m

```

% Created by Samira C. Oliva on 05/13/17
% Copyright (c) 2017 Samira C. Oliva. All rights reserved.
% Taylor Order 2 to solve IVP
function [t,w] = TaylorOrderTwo(a,b,N,alpha,fstring,fstring1)
h = (b-a)/N;
f = inline(fstring, 't', 'y'); %create function of diff. eq f(t_i,w_i)
f1 = inline(fstring1, 't', 'y'); %create function of f'(t_i,w_i)
t = linspace(a,b,N+1); %vector of mesh points
w = zeros(size(t)); %array of approximations of size same as t
w(1) = alpha;
for i=1:N %loop to compute the N remaining approximations
    term1 = f(t(i),w(i)); %evaluate f at t_i,w_i
    term2 = f1(t(i),w(i)); %evaluate f' at t_i,w_i
    w(i + 1) = w(i) + h*(term1 + (h/2)*term2); %difference equation
end
end

```

Runge_Kutta_OrderFour_method.m

```

% Created by Samira C. Oliva on 05/13/17
% Copyright (c) 2017 Samira C. Oliva. All rights reserved.
% Runge-Kutta order 4 to solve IVP
function [t,w] = Runge_Kutta_OrderFour_method(a,b,N,alpha,fstring)
h = (b-a)/N; %step size
f = inline(fstring, 't', 'y'); %create function of diff. eq f(t_i,w_i)
t = linspace(a,b,N+1); %vector of mesh points
w = zeros(size(t)); %vector of approximations
w(1) = alpha; %initial condition
for i=1:N %loop to compute the N remaining approximations
    k1 = h*f(t(i),w(i));
    k2 = h*f(t(i) + (h/2), w(i) + (1/2)*k1);
    k3 = h*f(t(i) + (h/2), w(i) + (1/2)*k2);
    k4 = h*f(t(i+1), w(i) + k3);
    w(i + 1) = w(i) + (1/6)*(k1 + 2*k2 + 2*k3 + k4); %difference equation
end
end

```

Live Script: programming_project4.mlx

% Created by Samira C. Oliva on 05/13/17

% Copyright (c) 2017 Samira C. Oliva. All rights reserved.

%main parameters

f = '2*y - exp(t)'; %diff_eq

f1 = '4*y - 3*exp(t)'; %derivative of diff_eq

a = 0;

b = 3;

alpha = 1;

N = 30;

%Eulers_method

[t,w] = Eulers_method(a,b,N,alpha,f);

plot(t,w,'g--');

hold on;

%Taylor order 2 Method

[t,w] = TaylorOrderTwo(a,b,N,alpha,f,f1);

plot(tv,yv,'b--');

hold on;

%Runge-Kutta 4

[t,w] = Runge_Kutta_OrderFour_method(a,b,N,alpha,f);

plot(t,w,'r--');

legend('Euler','TaylorO2','RK4');