

EE 3121 Differential Equations for Electrical Engineering

MiniProject #2

consider the differential equation

$$\frac{d^2 y}{dt^2} + 3 \frac{dy}{dt} + 2y = 0, \quad y(0) = 10, \dot{y}(0) = -2$$

Convert this second-order ODE into a system of two first-order ODEs, then solve it numerically for $0 < t < 4$

Let $x_1 = y$, $x_2 = \dot{y}$, then $\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$,

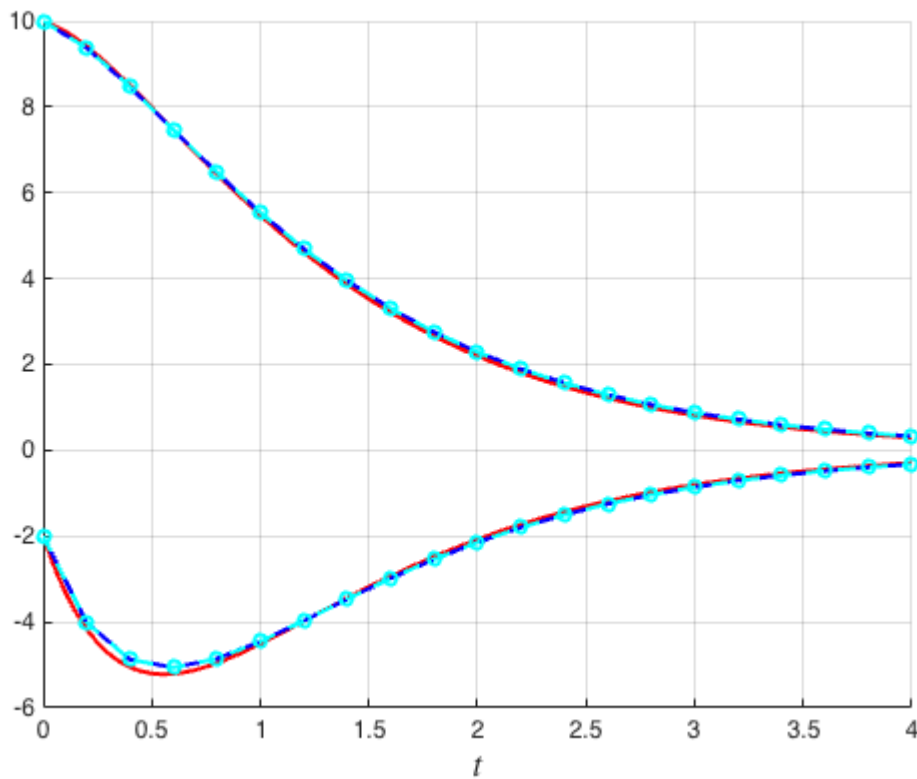
$$\frac{d\underline{x}}{dt} = \frac{d}{dt} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \dot{y} \\ -2y - 3\dot{y} \end{bmatrix} = \begin{bmatrix} x_2 \\ -2x_1 - 3x_2 \end{bmatrix} = \underline{f}(\underline{x})$$

```
f = @(t,x) ([x(2) ; -2*x(1)-3*x(2)]);  
x0 = [10 ; -2]; % initial value vector: [y(0) ; y'(0)]  
tspan = [0 , 4]; % time interval of solution: [initial time , final time]
```

analytical solution $y(t) = 18e^{-t} - 8e^{-2t}$

$$\underline{x}(t) = \begin{bmatrix} 18e^{-t} - 8e^{-2t} \\ -18e^{-t} + 16e^{-2t} \end{bmatrix}$$

```
[t_E, x_E] = euler_method(f, tspan, x0, 80);  
[t_RK, x_RK] = RK_method(f, tspan, x0, 20);  
[t_ode, x_ode] = ode45(f, tspan, x0); % Search "matlab ode" for details  
x_A = [18*exp(-t_RK)-8*exp(-2*t_RK) ; -18*exp(-t_RK)+16*exp(-2*t_RK)];  
figure()  
xlim([0 4])  
xlabel('$t$', 'Interpreter', 'latex', 'FontSize', 17)  
hold on  
grid on  
plot(t_E, x_E, Color='r', LineWidth=2)  
plot(t_RK, x_RK, Color='b', LineWidth=2)  
%plot(t_ode, x_ode, 'k:', LineWidth=2)  
plot(t_RK, x_A, 'co--', LineWidth=2)
```



Euler Method

$$\underline{x}(t_{n+1}) = \underline{x}(t_n) + \underline{f}(t_n, \underline{x}(t_n))h$$

```
function [t,x] = euler_method(f, tspan, x0, N)

ti = tspan(1); % initial time
tf = tspan(2); % final time
h = (tf-ti)/N; % step size
t = zeros(1,N+1); % pre-allocation
x = zeros (2,N+1); % pre-allocation
t(1) = ti; % initial value in array t
x(:,1) = x0; % initial values in matrix x

for k = 1:N
    s = feval(f, t(k), x(:,k));
    x(:,k+1) = x(:,k) + h*s;
    t(k+1) = t(k) + h;
end
end
```

Runge-Kutta Method

$$\underline{x}(t_{n+1}) = \underline{x}(t_n) + h \left(\frac{\underline{k}_{n1} + 2\underline{k}_{n2} + 2\underline{k}_{n3} + \underline{k}_{n4}}{6} \right)$$

where

$$\begin{aligned}\underline{k}_{n1} &= \underline{f}(t_n, \underline{x}(t_n)), \\ \underline{k}_{n2} &= \underline{f}(t_n + h/2, \underline{x}(t_n) + (h/2)\underline{k}_{n1}), \\ \underline{k}_{n3} &= \underline{f}(t_n + h/2, \underline{x}(t_n) + (h/2)\underline{k}_{n2}), \\ \underline{k}_{n4} &= \underline{f}(t_n + h, \underline{x}(t_n) + h\underline{k}_{n3}).\end{aligned}$$

```
function [t,x] = RK_method(f, tspan, x0, N)

ti = tspan(1); % initial time
tf = tspan(2); % final time
h = (tf-ti)/N; % step size
t = zeros(1,N+1); % pre-allocation
x = zeros (2,N+1); % pre-allocation
t(1) = ti; % initial value in array t
x(:,1) = x0; % initial values in matrix x

for k = 1:N
    k1 = feval(f, t(k), x(:,k));
    k2 = feval(f, t(k)+h/2, x(:,k)+k1*h/2);
    k3 = feval(f, t(k)+h/2, x(:,k)+k2*h/2);
    k4 = feval(f, t(k)+h, x(:,k)+k3*h);
    x(:,k+1) = x(:,k) + (k1 + 2*k2 + 2*k3 + k4) * h/6;
    t(k+1) = t(k) + h;
end
end
```