Given a circuit with the following transfer function

\[ H(s) = \frac{s+1}{s^2+2s+3} \]

with initial conditions \( y(0^+) = 4, y'(0^+) = -2 \). Use numerical integration to find the circuit output \( y(t) \) when the input is \( x(t) = t e^{-2t} u(t) \).

Note: The Laplace solution method as described in class uses the initial conditions at \( t = 0^- \). The numerical integration method requires the initial conditions at \( t = 0^+ \). In general, the initial conditions at \( t = 0^- \) are different than those at \( t = 0^+ \). If the desired output is an inductor current or capacitor voltage they may be the same (they can be different even then if the input or one of the derivatives of the input is an impulse). For many other outputs the initial conditions will usually be different.

**Method**

Use the transfer function to find the corresponding differential equation relating the output \( y(t) \) to the input \( x(t) \):

\[ \frac{d^2 y}{dt^2} + 2 \frac{dy}{dt} + 3 y = \frac{dx}{dt} + x \]

Octave/MATLAB require that the differential equation be written as a first-order system of differential equations in the form:

\[ y' = f(t, y) \]

where \( y' \) and \( y \) are vectors. There is a standard trick for converting a differential equation into this form. Let \( y_1(t) = y(t) \) and \( y_2(t) = y'(t) = y'(t) \). Then \( y_2(t) = y''(t) \). Solving the differential equation above for \( y''(t) \) we have:

\[ y'' = x' + x - 2 y' - 3 y \]

or

\[ y_2' = x' + x - 2 y_2 - 3 y_1 \]

For all \( t > 0 \), \( x(t) = t e^{-2t} \) then \( x'(t) = e^{-2t} (1 - 2t) \) and \( x'(t) + x(t) = e^{-2t} (1 - t) \) so we can write:

\[ y_1' = y_2 \]
\[ y_2' = e^{-2t} (1 - t) - 2 y_2 - 3 y_1 \]

This is a system of equations in the required form. We can numerically solve the differential equation and plot the result in only four lines of Octave/MATLAB code:
\[ y_0 = [4; -2]; \]
\[ ydot = @(t, y) [y(2); \exp(-2*t)*(1 - t) - 2*y(2) - 3*y(1)]; \]
\[ [t y] = \text{ode45}(ydot, [0 8], y0); \]
\[ \text{plot}(t, y(:,1)) \]

This produces the graph below:

The first line of code sets the initial conditions. The second line defines a function (ydot) that corresponds to the first-order system of equations presented previously. The function takes arguments \( t \) and \( y \) where \( y \) is a column vector with two rows. The function must also return a two-row column vector. The first row of the return vector is \( y(2) \) while the second row contains \( \exp(-2*t)*(1 - t) - 2*y(2) - 3*y(1) \). Compare this to the system of equations on the bottom of the previous page. The third line of code uses the Octave/MATLAB routine `ode45` to numerically solve the system of equations. The routine takes as arguments the ydot function, a vector containing the beginning and ending time points over which we want the solution and a column vector containing the initial conditions. It returns a column vector \( t \) containing the times at which the solution is computed and a matrix \( y \) containing the solution. The first column of \( y \) contains the solution values for \( y_1(t) \) while the second column contains the solution values for \( y_2(t) \). We are only interested in \( y_1(t) \) here since that corresponds to our desired \( y(t) \). The fourth line of code plots the first column of \( y \) (\( y(:,1) \)) versus the corresponding time vector.