**Lab 5: Inverse Laplace Transform via MATLAB**

**Objective:**

The Laplace Transform converts the integral-differential equations of electromechanical system into algebra-based equations that can be easily manipulated. However, evaluating the inverse Laplace transform can be cumbersome. This lab will introduce some tools in MATLAB that can be used to find the inverse Laplace transform.

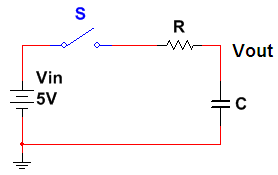


Figure 5.1: Series RC circuit to be analyzed

**Pre-Lab:**

Read section 13.2 from Nilsson & Riedel and understand how to convert R, L and C in time domain into s-domain, as shown below.

|  |  |  |
| --- | --- | --- |
| Parameter | Time Domain | s-domain |
| Voltage | v(t) | V(s) |
| Current | i(t) | I(s) |
| Resistance | R | R |
| Inductance | L | sL |
| Capacitance | C | 1/sC |

**Procedure:**

In Figure 5.1, select your own values of *R* and *C*.

The switch closes at t = 0. The output (*Vout*) is the capacitor voltage.

Once converted to frequency domain, all components can be treated like “resistors”. Treat all components like “resistors” and solve for *Vout(s)* in the Laplace domain.

The voltage divider gives .

The time-domain solution *Vout(t)* requires us to find the inverse Laplace transform *Vout(s).*

1. Find the inverse Laplace transform using the following 2 methods:

**Method 1**: Using the MATLAB built-in function residue

Let the denominator be  and the numerator be

Then the MATLAB function [r, p, K] = residue(b, a) finds the partial fraction expansion:



Once the partial fraction expansion is obtained, you can write down the inverse Laplace transform:



Here we assume that we have only simple poles. When the order of the numerator *b* is lower than the order of the denominator *a*, we always have *K* = 0.

**Method 2**: Using the MATLAB’s symbolic calculation and function ilaplace

As an exercise, run the following MATLAB script to learn about MATLAB’s laplace and ilaplace :

syms t %time variable t

f=2\*exp(-t)-2\*t\*exp(-2\*t)-2\*exp(-2\*t); %define f(t)

pretty(f) %looks better

F=laplace(f) %Laplace transform

pretty(F) %looks better

F=simplify(F) %combine partial fractions

fnew=ilaplace(F) %inverse Laplace transform

pretty(f) %looks better

Now you are ready to do the lab using the second method.

i. Defining the symbolic variables to be used (i.e. *s*)

>> syms s

ii. Writing the Laplace domain function

>> F = b/(R\*C\*s^2 + s)

iii. Operating on the function

>> f = ilaplace(F)

**Post-Lab:**

1. Did the two methods give you the same mathematical expression for the inverse Laplace transform?
2. Type (or write) the two time-domain expressions in your conclusion.
3. Run a *Multisim* simulation to verify the time-domain *Vout*(*t*) is reasonably correct.

**Appendix**

Suggested MATLAB code: Please change your values of R & C

clc; %reset the workspace command line

clear all; %clear all the variables

close all; %close all the plots

%%========Define the circuit==========

%%NOTE: Please use your own value for R and C

R = 10000; %10kohm – CHANGE to Your value

C = 0.1\*10^(-6); %0.1 uF – CHANGE to Your value

vin = 5; %input amplitude=5 V

a = [R\*C 1 0]; %denominator

b = vin; %numerator

%%=========For Plotting==================

set(gca,'fontsize',18,'FontWeight','bold','FontName','Times New Roman');

%%=======================================

%% Method 1: Residue

display('Method1: Residue');

[r, p, K] = residue (b, a)

t=0:0.0001:0.01;

VB=r(1)\*exp(p(1)\*t)+r(2)\*exp(p(2)\*t);

subplot(1,2,1)

plot(t, VB)

xlabel('Time[s]');

ylabel('Voltage [V]');

title('Method 1: Residue')

%%%===================================

% Method 2: Symbolic

display('Method2: Symbolic');

syms s

F = b/(a(1)\*s^2+a(2)\*s)

f = ilaplace(F)

subplot(1,2,2)

ezplot(f, [0, 0.01]);%ezplot plots function f over the specified range

xlabel('Time[s]');

ylabel('Voltage [V]');

title('Method 2: Symbolic')