ECE 3620
Programming Assignment # 2

Numerical Solution of Differential Equations: The Total Solution

Due: November 15, 2022

Objective and Background

This assignment has four basic objectives:

1. To continue to remind you that you know how to program in C or C++.
2. To give you some practice in converting a mathematical description of an idea into a computer implementation of the idea.
3. To provide practice and experience in convolution.
4. To provide a little more practice in finding the complete solution of differential equations.

In this assignment you will write a program in C or C++ to perform numerical convolution of causal signals through causal systems.

As you recall, the formula for continuous-time convolution is given by

$$y(t) = \int_0^t f(\tau)h(t-\tau)d\tau$$

This was arrived at by taking the following limit

$$y(t) = \lim_{\Delta\tau \to 0} \sum_{m=0}^{\infty} f(m\Delta\tau)h(t - m\Delta\tau)\Delta\tau \tag{1}$$

If we define $\Delta\tau$ as the sampling interval T, (1) can be written as

$$y(nT) = T\sum_{m=0}^{n} f(mT)h(nT - mT)$$

We can change notation to indicate sampled signals by realizing that $y(nT)$ can be written $y[n]$. This results in the sum

$$y[n] = T\sum_{m=0}^{n} f[m]h[n-m]$$

The initial part of this programming assignment is to write a function which will compute the **convolution sum** of two sequences $f[n]$ and $h[n]$. The convolution sum is defined as

$$y[n] = \sum_{m=0}^{n} f[m]h[n-m] \equiv f[n] * h[n] \tag{2}$$

Note that the sum is *not* scaled by T.

Assignment

For the following exercises, **turn in your programs, your analytical computations, your plots, and your comments and observations, as appropriate**. The plots should be clearly labeled.

To practice preparing a professional report, include the above information in a single .pdf document with a title, sections containing the problems, and concluding comments and observations. Your handwritten work should be scanned and included in the document.

The program should be written in C or C++. Only libraries from the C/C++ standard libraries are permitted. No other external libraries should be used – you are writing the entire program from scratch. You must write all of the code necessary for the processing.

1. Write a subroutine (function) that will convolve two sequences using (2). You might want to use a function description like

   ```
   leny = conv(double *f1, int len1, double *f2, int len2, double *y)
   ```

   which would convolve the sequence in the array `f1` having `len1` points with the sequence in the array `f2` having `len2` points. The result of the convolution is returned in the array `y`, and the number of points in the convolution is returned in `leny`.

2. Test your program by convolving the following functions:

   (a) $f_1 * f_1$

   (b) $f_1 * f_2$

   (c) $f_1 * f_3$

   (d) $f_2 * f_3$

   (e) $f_1 * f_4$

   where the sequences are described by the C/C++ arrays

   ```
   f1[] = {0,1,2,3,2,1};
   len1 = 6;
   f2[] = {-2,-2,-2,-2,-2,-2,-2};
   len2 = 7;
   f3[] = {1,-1,1,-1};
   len3 = 4;
   f4[] = {0,0,0,-3,-3};
   len4 = 5;
   ```

   Remember that C/C++ arrays start with an index of 0.

   Plot these functions. Verify that the convolution is working as it should.

   Do the convolutions by hand. Also, compute the same convolutions using the `Matlab` function `conv`. Compare the results from the three methods (they better all be the same!).

3. Using the results from Program #1 (which computed the zero-input response), find the **total solution** to the differential equation

   $$(D^3 + 3D^2 + 38.25D + 72.5)y(t) = (D + 1.5)f(t)$$

   with initial conditions $y(0) = -2$, $\dot{y}(0) = 3$, $\ddot{y}(0) = 4$ and $f(t) = \sin(2.5\pi t)u(t)$. Use $T = 0.001$, and let $0 \leq t \leq 10$. You will need to incorporate part of Program #1 into your new program to complete this part.

   (a) Using paper-and-pencil analysis, find the impulse response of the system $h(t)$. Then compute and plot its sampled values $h[k] = h(kT)$. You may use Laplace transform methods if you want. You may also use Matlab to find the poles.

   (b) Find the sampled values of the input function $f[k] = f(kT)$. Plot these values.

   (c) The zero-state solution is the scaled convolution $T(f[k] * h[k])$.

(d) The zero-input solution is found using Program #1.

(e) The total solution is the sum of the zero-state solution and the zero-input solution. (You will somehow need to incorporate the data computed using code from Program #1 with the data from this program to get the total solution.)

(f) Compute **numerically** and plot the total solution.

(g) Find an **analytical** solution to the DE and plot it.

(h) Compare the analytical and the numerical solution. (Comment)

4. Note that the solution to the system output in 3 above settles to a "steady-state" solution after a few seconds, where the signal form continues unchanged. Your plots should indicate this. What is the steady-state output amplitude? Why does this happen?

Hints and helps

Limits As for many convolution problems, the hardest part of the convolution is getting the limits of summation correct. Pay very close attention to the upper and lower limits of summation.

Number of points Also pay close attention to the number of points which appear in the convolved output.

Two nested for loops The basic structure for convolution is simple:

```
for(k = 0; k <= upper limit on k; k++) {
   sum = 0;
   for(m = lower limit on m; m <= upper limit on m; m++) {
      sum += f1[m]*f2[k-m];
   }
   y[k] = sum;
}
```

All you really need to do (which will require some care and understanding; you are are on your own for this!) is to find the lower and upper limits of the two `for` loops.