Program 2 Report

Kelson Petersen

November 14, 2025

Objectives

To review programming in C or C++. To give practice in converting mathematical descriptions of ideaas into computer implementations. To provide practice and experience in convolution. To provide practice in finding the complete solution of differential equations.

Discrete Convolution in C

To perform convolution in a digital computer it is easier to make a continuous signal discrete. We do this by 'sampling' the continuous signal at specific points in time. We then are able to convolve that signal with other discrete signals. The method to convolve two discrete functions is create an array that holds $n \cdot m - 1$ elements where n is the length of signal one, and m is the length of signal two. The convolution is the sum of all diagonal elements in a table of products as seen in the scratch work section of the appendix.

To write this code it is fairly simple and only requires the following code.

```
1  #include "convolve.h"
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  int conv(double *f1, int len1, double *f2, int len2, double **y){
6    int leny = len1 + len2 - 1;
7    int in_b, m_start;
8    // Allocate The proper amount of memory for y
9    (*y) = (double *)malloc((sizeof(double) * leny));
10   for(int itr = 0; itr < leny; itr++){
11     (*y)[itr] = 0.0f;
12   }
13   // Add the diagonals
14   for(int i = 0; i < len1; i++){
15     for(int j = 0; j < len2; j++){
16       (*y)[i+j] += f1[i] * f2[j];
17     }
18   }
19   return leny;
20 }
```

The double nested loop in lines 14–18 iterate through each element of the two arrays passed to the function. While iterating throught the arrays the index $i + j$ is how we get the diagonal elements to sum together in the correct places.

Conclusion

Words

Appendix

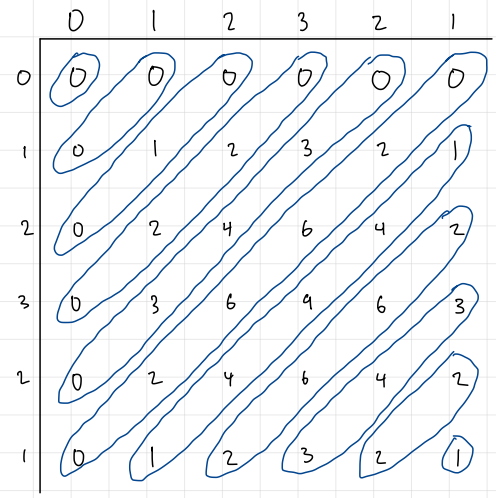
Words

Code

Words

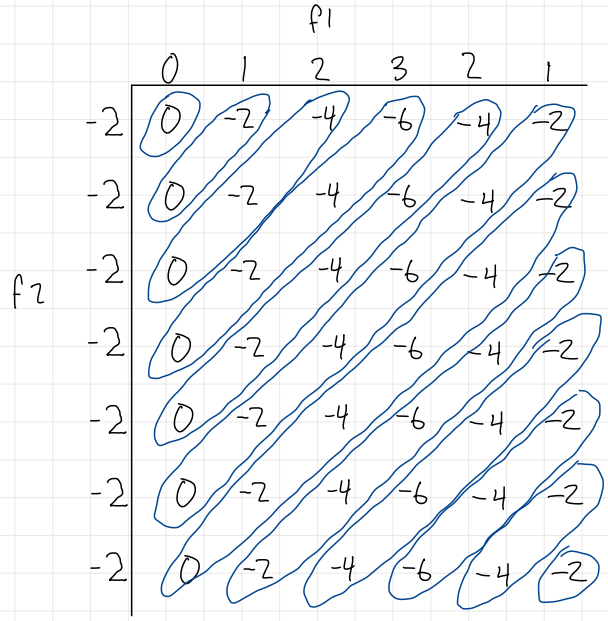
Scratch Work

2.A.) $f1 * f1 = 0, 0, 1, 4, 10, 16, 19, 16, 10, 4, 1$



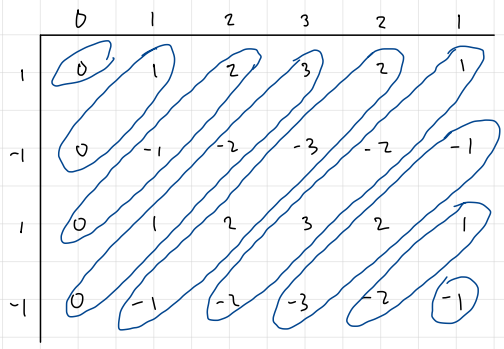
m	y[m]	
0	0	= 0
1	0+0	= 0
2	0+1+0	= 1
3	0+2+2+0	= 4
4	0+3+4+3+0	= 10
5	0+2+6+6+2+0	= 16
6	1+4+9+4+1	= 19
7	2+6+6+2	= 16
8	3+4+3	= 10
9	2+2	= 4
10	1	= 1

2.B.) $f1 * f2 = 0, -2, -6, -12, -16, -18, -18, -18, -16, -12, -6, -2$



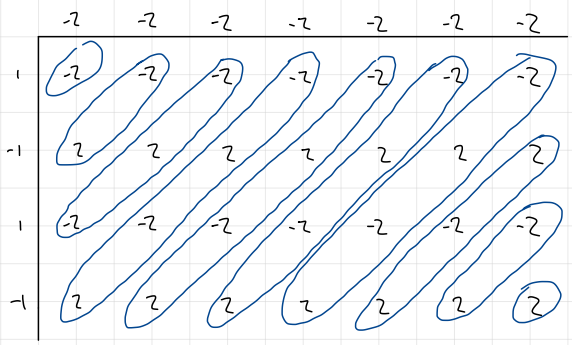
m	y[m]	
0	0	= 0
1	0-2	= -2
2	0-2-4	= -6
3	0-2-4-6	= -12
4	0-2-4-6-4	= -16
5	0-2-4-6-4-2	= -18
6	0-2-4-6-4-2	= -18
7	-2-4-6-4-2	= -18
8	-4-6-4-2	= -16
9	-6-4-2	= -12
10	-4-2	= -6
11	-2	= -2

2.C.) $f1 * f3 = 0, 1, 1, 2, 0, 0, -2, -1, -1$



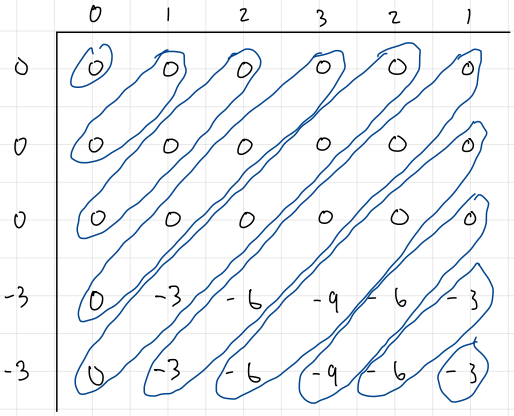
k	y[k]	
0	0	= 0
1	0+1	= 1
2	0-1+2	= 1
3	0+1-2+3	= 2
4	-1+2-3+2	= 0
5	-2+3-2+1	= 0
6	-3+2-1	= -2
7	-2+1	= -1
8	-1	= -1

2.D.) f2 * f3 = -2, 0, -2, 0, 0, 0, 0, 2, 0, 2



k	y[k]	
0	-2	= -2
1	2-2	= 0
2	-2+2-2	= -2
3	2-2+2-2	= 0
4	2-2+2-2	= 0
5	2-2+2-2	= 0
6	2-2+2-2	= 0
7	2-2+2	= 2
8	2-2	= 0
9	2	= 2

2.E.) f1 * f4 = 0, 0, 0, 0, -3, -9, -15, -15, -9, -3



k	y[k]	
0	0	= 0
1	0+0	= 0
2	0+0+0	= 0
3	0+0+0+0	= 0
4	0-3+0+0+0	= -3
5	-3-6+0+0+0	= -9
6	-6-9+0+0	= -15
7	-9-6+0	= -15
8	-6-3	= -9
9	-3	= -3