Mason Nixon
ELEC 5410
Dr. Stanley Reeves
October 8, 2010

Project #4: The DFT and FFT

1. Write a MATLAB function that implements a DFT using the DFT summation formula. Include a listing of your function.

```
function X = DFT(sequence)
N = length(sequence);
Xi = 0; X = 0;
for k = 1:N;
    for n = 1:N;
         Xi(n) = sequence(n)*exp(-j*2*pi*(n-1)*(k-1)/N);
    end
X(k) = sum(Xi);
End
```

2. The overhead associated with the loops and the calculation of the multipliers in a DFT is considerable in MATLAB. To eliminate this from the process, create a matrix F that can be used to multiply a column vector to perform a DFT:

```
F = dftmtx(2^11);
```

Create a random sequence N = 2^{11} points long. Perform a DFT by multiplying the sequence (as a column vector) by the DFT matrix. Report the time required for the DFT matrix multiply. Do not include the time required to create the DFT matrix. Use `tic` and `toc` or `cputime` to time this operation. (To get accurate timing, before you begin, create the output variable by doing something like:

```
Xf = zeros(2^11,1);
```

Compare to the time required for calculating the DFT of the same sequence using the function you wrote.

The time required to perform the DFT using my summation implementation was 2.0592 seconds, whereas the time taken to DFT using matrix multiplication was approximately 0.0156 seconds according to the `cputime` and `tic` and `toc` functions in MATLAB. This is most likely because MATLAB is an interpreted language and it must interpret every line in a for loop each time it goes through the loop. Since the second method uses matrix multiplication (and since MATLAB is designed for vector operations), the DFT is calculated 42.3 times faster.

3. Use `fft` to compute the DFT of the same sequence and report the time. (Do the FFT in a loop 10,000 times and average the resulting time.)

The `fft` function calculates the DFT in 39 microseconds after averaging 10,000 iterations.

4. Determine the time required to compute an FFT in MATLAB of a sequence with length 2^{11} - 9. Explain why this takes longer than a length-2^{11} FFT.

The `fft` function calculates the DFT in 371.28 microseconds after averaging 10,000 iterations. According to Mathworks, "If *n* is a power of 2, a one-dimensional FFT of length *n* requires less than $3n \log_2 n$ floating-point operations (times a proportionality constant)," and "The execution time of an FFT algorithm depends on the transform length. It is fastest when the transform length is a power of two, and almost as fast when the transform length has only small prime factors. It is typically slower for transform lengths that are prime or have large prime factors." Since the sequence length is not a power of 2 and it is a prime number, i.e. the standard algorithm cannot be used and the processing time is much longer than it would be otherwise.

5. Write a MATLAB function that implements a linear convolution of any two input sequences of any length using power-of-two FFT's. (You can call the MATLAB `fft` function with an optional argument that specifies the sequence length; specify a length that is equal to a power of two.) Include a listing of your function. Generate two sequences of length 2^{12}. (It doesn't matter what they are.) Convolve the two sequences with your function and with `conv`. Compare the result to make sure they're the same, and compare the time required to run them. (You may need to accumulate the time required for 1000 runs and then average.)

```
function output = num5(x1,x2)
    N1 = length(x1);
    N2 = length(x2);
    N3 = N1+N2-1;
    t4=0;
    for a=1:1000
        t=cputime;
        prod=fft(x1,N3).*fft(x2,N3);
        myconv=ifft(prod);
        t4=t4+cputime-t;
    end
    tfftconv=t4/1000;
    t5=0;
    for a=1:1000
        t=cputime;
        mconv=conv(x1,x2);
        t5=t5+cputime-t;
    end
    tconv=t5/1000;
    output={myconv, tfftconv, mconv, tconv};
end
```

Figure1

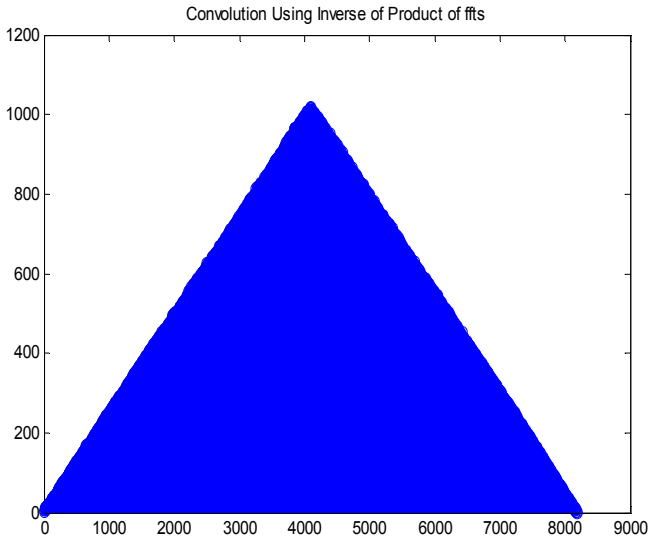
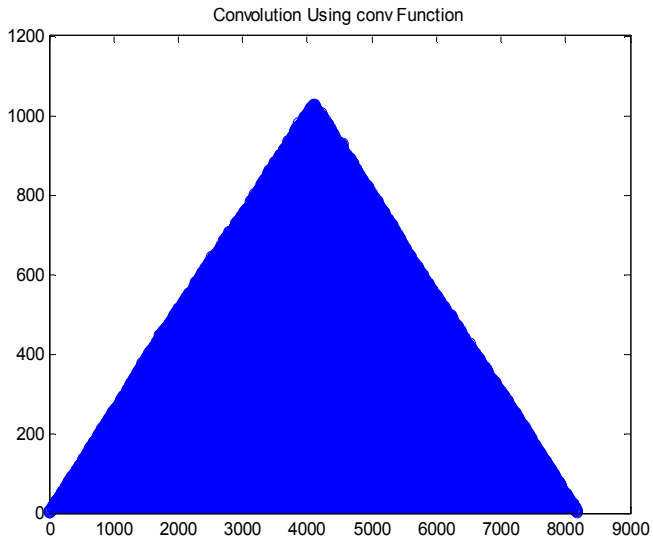


Figure2



The average of 1000 time trials taken by the inverse of the product of FFTs is 0.00655 seconds, while the time taken for the convolution is 0.03031 seconds.

6. 6410 or extra credit for 5410: Write a MATLAB function that implements a non-power-of-two FFT using power-of-two FFT's, according to the method discussed in class in which the FFT is transformed into a convolution problem. You may use the MATLAB FFT to implement the required convolution, as long as it is called so that it computes a power-of-two FFT. Compare the timing on a random sequence of length 999,983 to the MATLAB FFT. Include a listing of your function.

```
function output = num6(x)
    N = nextpow2(length(x));
    t6=0;
    for a=1:1000
        t=cputime;
        myfft=fft(x,2^N);
        t6=t6+cputime-t;
    end
    tfft=t6/1000;

    t7=0;
    for a=1:1000
        t=cputime;
        mfft=fft(x);
        t7=t7+cputime-t;
    end
    tmfft=t7/1000;
    output={myfft,t6,mfft,t7};
end
```

The average of 1000 time trials taken by the power-of-two method is 0.29688 seconds, while the non power-of-two FFT took 2.1875 seconds. The power-of-two method proves to be 7.37 times faster.