AMATH 341 / CS 371 Fall 2004: Assignment 4

Instructor: Hans De Sterck Office: MC 5016 e-mail: *hdesterck@uwaterloo.ca*
TA: Jenny Lee Office: MC 5133 e-mail: *j39lee@math.uwaterloo.ca*

Web Site: http://www.math.uwaterloo.ca/~hdesterc/websiteW/courses/amath341.html
Newsgroup: uw.cs.cs371

Due Date: Friday November 19, beginning of class

1. (10 marks)

   (a) Find the Fourier Transform $F(q)$ of

   $$f(t) = \frac{A\ t}{L} \quad \text{for} \quad t \in [-L, L]; \qquad f(t) = 0 \quad \text{elsewhere.}$$

   Sketch $f(t)$ and $F(q)$.

   (b) Find the Fourier Series Expansion of

   $$f_T(t) = \frac{t}{\pi} \quad \text{for} \quad t \in [0, 2\pi] \quad (T = 2\pi).$$

   Give the result both in complex form and in real form.

2. (10 marks)
   Calculate the Discrete Fourier Transform of the following periodic time sequences by hand, both using the direct DFT formula, and the FFT algorithm. For the FFT calculations, organize your work in a butterfly diagram.

   (a)

   $$f[n] = (1, 2, 2, 1) \quad (n = 0, \ldots, 3;\ N = 4)$$

   (b)

   $$f[n] = (1, 2, 3, 2) \quad (n = 0, \ldots, 3;\ N = 4)$$

   Why is the resulting transform real in this case?

3. (15 marks)

   (a) Write a Matlab function that implements the DFT using the direct formula. The first line of your Matlab m-file `slowFT.m` should read
   `function DFT=slowFT(f,N).`
   Here f is a complex vector of length N that contains the input time series, and DFT is the resulting complex DFT.

(b) Write a Matlab function that implements the DFT using the FFT algorithm. You can base your implementation on the pseudocode given in the course notes. The first line of your Matlab m-file `fastFT.m` should read
`function DFT=fastFT(f,N).`
Again, f is a complex vector of length $N = 2^m$ that contains the input time series, and DFT is the resulting complex DFT.

(c) Compare the performance of your slowFT and fastFT implementations. Generate random time series $f = rand(N, 1)$ with $N = 2^k$, for suitable values of k. For each of the methods, see how large you can make k and still get an acceptable turnaround time. Tabulate execution times (you can use the `clock` and `etime` commands) and verify that the complexities are approximately $O(N^2)$ and $O(N \log(N))$ as theoretically expected. (Note: execution times will depend on machine load, memory structure and availability, etc., so don't expect that the theoretical complexity predictions will hold exactly in practice! Performance optimization of your code is optional. For example, assigning the complex roots of unity W_N takes a lot of time in the algorithm, if implemented naively. A lot of time can be saved by calculating these values once and saving the results in a table. Vectorization is also essential in order to get good performance.)

See the Note at the end of this assignment about how to hand in your code.

4. To Be Continued ... (more questions will be added soon)

5.

...

Note: For Question 3, place your versions of slowFT.m and fastFT.m in a single directory (nothing else in this directory) with the name `your_email _your_student_id`. Then zip up this directory (using zip -r `your_email_your_student_id your_email_your_student_id` if on a unix machine). Mail the file `your_email_your_student_id.zip` to j39lee@math.uwaterloo.ca by 1:30 pm on the due date. Your code will be subject to black box testing.