CS 621: Assignment 1

Due: Wednesday, September 15, 2004 (By 4pm in Upson 4130)

Scoring on each problem is on a 0-1-2-3 scale. 3 = complete success, 2 = overlooked a small point, 1 = germ of the right idea, 0 = missed the point of the problem. One point will be deducted for insufficiently commented code. Test drivers and related material will be posted on the course website `http://www.cs.cornell.edu/courses/cs621/2004fa/`. For each problem submit output and a listing of all the scripts/functions that you had to write/modify in order to produce the output. Solutions should always be vectorized, that is, you must fully exploit *Matlab*'s vector operation capabilities. You are allowed to discuss *background* issues with other students, but the codes you submit must be your own.

Problem 1. (First Column of a Hessenberg Product)

The matrix $H \in \mathbb{R}^{n\times n}$ is said to be *Hessenberg* if it has lower bandwidth 1, e.g,

$$H = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}.$$

Think about the structure of the vector

$$v = (H - r_p I_n)\cdots(H - r_1 I_n)e_1 \qquad r \in \mathbb{R}^p,\ p < n$$

where e_1 is the first column of I_n and complete the following MATLAB function so that it performs as specified:

```
  function y = FirstCol(H,r)
% H is an n-by-n matrix with lower bandwidth 1.
% r is a column p-vector with p < n
% y is the first column of (H - r(p)I)*(H - r(p-1)I)*...*(H - r(1)I)
```

Your implementation should be flop efficient. Test your implementation with the script `P1` which is available on the course website.

Problem 2. (Strassen Multiplication)

Refer to the description of the Strassen matrix multiplication algorithm for $C = AB$ on page 31-33 of GVL. This recursive algorithm involves fewer flops than the conventional method. In this problem you explore the improvement as a function of the recursion base parameter n_{min}. In particular, complete the following MATLAB function so that it performs as specified:

```
  function nFlops= StrassFlops(n,nmin)
% n and nmin are powers of two and nmin < n
% nFlops is the number of flops required when Strassen's
% algorithm is applied to an n-by-n matrix multiply problem
% with recursion base parameter nmin.
```

Test your implementation by running the script `P2` which is available on the course website. *Extra Point:* My solution suggests that $n_{min} = 16$ is optimal. Can you explain why?

Problem 3. (DFT Matrix Set-up)

The n-by-n DFT (discrete Fourier transform) matrix $F_n = (f_{pq})$ is defined by $f_{pq} = \omega_n^{(p-1)(q-1)}$ where $\omega_n = \cos(2\pi/n) - i\sin(2\pi/n)$. For example,

$$F_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 \\ 1 & \omega^2 & \omega^4 & \omega^6 & \omega^8 \\ 1 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} \\ 1 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} \end{bmatrix}, \qquad \omega = \omega_5$$

Complete the following function so that it performs as specified:

```
  function F = dft(n)
% F is the n-by-n DFT matrix
  theta = 2*pi*(0:n-1)'/n;
  w = cos(theta) - sqrt(-1)*sin(theta);
```

Your implementation should *not* involve any additional trig evaluations. Vectorize as much as possible. Test your implementation by running the script `P3` which is available on the course website. For your information, one never explicitly forms the DFT matrix in practice. Matrix vector products that involve F_n or F_n^{-1} can be evaluated in $O(n \log n)$ time using the MATLAB functions `FFT` and `IFFT`. (You'll be using them in the next problem.)

Problem 4. (Fast Toeplitz Multiplication)

A matrix $T \in \mathbb{R}^{n \times n}$ is *toeplitz* if it is constant along all its diagonals, e.g.,

$$T = \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ c_2 & r_1 & r_2 & r_3 \\ c_3 & c_2 & r_1 & r_2 \\ c_4 & c_3 & c_2 & r_1 \end{bmatrix}$$

Note that if $c, r \in \mathbb{R}^n$ and $r_1 = c_1$ then `T = toeplitz(c,r)` is a how to set this matrix up in MATLAB.

A matrix $C \in \mathbb{R}^{n \times n}$ is *circulant* if it is Toeplitz and each column is the "downshigted" version of its predecessor, e.g.,

$$C(h) = \begin{bmatrix} h_1 & h_4 & h_3 & h_2 \\ h_2 & h_1 & h_4 & h_3 \\ h_3 & h_2 & h_1 & h_4 \\ h_4 & h_3 & h_2 & h_1 \end{bmatrix}$$

We refer to $h \in \mathbb{R}^n$ as the *circulant vector*. Consider the following facts:

- If $T \in \mathbb{R}^{n \times n}$ is Toeplitz then there exists a circulant matrix $C \in \mathbb{R}^{m \times m}$ where m is a power of two and $C(1{:}n, 1{:}n) = T$.
- Suppose $T \in \mathbb{R}^{n \times n}$, $C \in \mathbb{R}^{m \times m}$, $T = C(1{:}n, 1{:}n)$ and $x \in \mathbb{R}^m$ with $x(n+1{:}m) = 0$. If $y = Cx$ then $y(1{:}n) = Tx(1{:}n)$.
- If $C(h) \in \mathbb{R}^{m \times m}$ is circulant then
$$C(h) = F_n^{-1} \mathrm{diag}(F_n h) F_n$$
If $x \in \mathbb{R}^n$ and $h \in \mathbb{R}^n$ are given and $\tilde{x} = F_n x$ and $\tilde{h} = F_n h$, then $y = C(h)x = F_n^{-1}\left(\tilde{h}. * \tilde{x}\right)$. Suppose m is a power of two. If the FFT is used to compute $\tilde{x}$ and $\tilde{h}$ and the inverse FFT is applied to their pointwise product to get y, then y can be obtained in $O(m \log m)$ flops.

Using these facts, develop a fast implementation of the following MATLAB function:

```
  function y = FastToep(c,r,x)
% r and c are column n-vectors with r(1) = c(1).
% x is a column n-vector
% y = Toeplitz(c,r)*x
```

Test your implementation by running the script `P4` which is available on the course website. Note that if x is real then so is $y = Tx$, but that complex numbers arise during the solution process. When you are all done y might have a small imaginary part. These can be safely ignored.