# CS 4220: Assignment 1

Due: Monday, February 8, 2010 (In Class or in Upson 5153 by 4pm)

Scoring for each problem is on a 0-to-5 scale ( 5 = complete success, 4 = overlooked a small detail, 3 = good start, 2 = right idea, 1 = germ of the right idea, 0 = missed the point of the problem.) One point will be deducted for insufficiently commented code. Unless otherwise stated, you are expected to utilize fully MATLAB's vectorizing capability subject to the constraint of being flop-efficient. Test drivers and related material are posted on the course website `http://www.cs.cornell.edu/courses/cs4220/2010sp/`. For each problem submit output and a listing of all scripts/functions that *you* had to write in order to produce the output. You are allowed to discuss *background* issues with other students, but the codes you submit must be your own.

### P1. (A Structured Matrix Multiplication)

Symmetry and block structure are very important in matrix computations. This problem deals with both. Complete the following MATLAB function so that it performs as specified.

```
  function C = MatSquare(H)
% H is a 2n-by-2n matrix whose entries satisfy
%          (i)   H(1:n,1:n) = -H(n+1:2n,n+1:2n)'.
%          (ii)  H(1:n,n+1:2n) = H(1:n,n+1:2n)'.
%          (iii) H(n+1:2n,1:n) = H(n+1:2n,1:n)'.
% C = H*H.
```

The matrix $H$ and $H^2$ are structured. Your implementation should strive to minimize flops by exploiting this structure. Make full use of MATLAB's vectorizing capability. Build intuition for the problem by examining small-$n$ examples. Submit a listing of your implementation together with the output that is produced when the test script `P1` is applied. A successful implementation involves no loops and $8n^3$ flops.

*Review:* `norm`, `function`, `randn`, `size`.

### P2. (Interpolation with Chebyshev Polynomials)

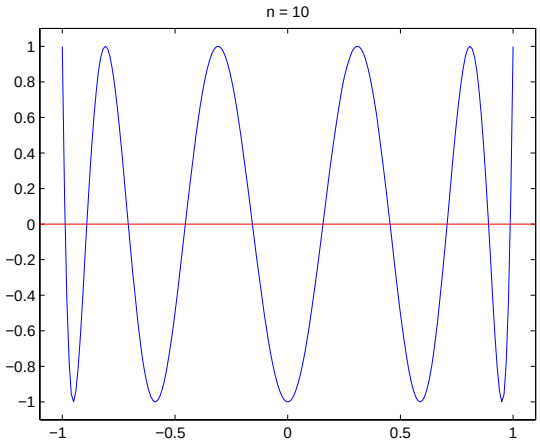
The Chebyshev polynomials $T_0(x)$, $T_1(x)$, $T_2(x)$ are defined by

$$T_n(x) \;=\; 2xT_{n-1}(x) - T_{n-2}(x)$$

where $T_0(x) = 1$ and $T_1(x) = x$. Thus,

$$\begin{aligned} T_2(x) &= 2xT_1(x) - T_0(x) = 2x^2 - 1 \\ T_3(x) &= 2xT_2(x) - T_1(x) = 2x(2x^2-1) - x = 4x^3 - 3x \end{aligned}$$

and so on. These polynomials oscillate betwwen -1 and +1 on the interval [-1 1], e.g.,



Suppose we are given points $(x_1, y_1)$, $(x_2, y_2)$, $(x_3, y_3)$, and $(x_4, y_4)$. Assume that the $x_i$ are distinct. Convince yourself that if we solve the linear system

$$\begin{bmatrix} T_0(x_1) & T_1(x_1) & T_2(x_1) & T_3(x_1) \\ T_0(x_2) & T_1(x_2) & T_2(x_2) & T_3(x_2) \\ T_0(x_3) & T_1(x_3) & T_2(x_3) & T_3(x_3) \\ T_0(x_4) & T_1(x_4) & T_2(x_4) & T_3(x_4) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

and define the cubic polynomial

$$p_3(x) \;=\; \alpha_1 \cdot T_0(x) + \alpha_2 \cdot T_1(x) + \alpha_3 \cdot T_2(x) + \alpha_4 \cdot T_4(x)$$

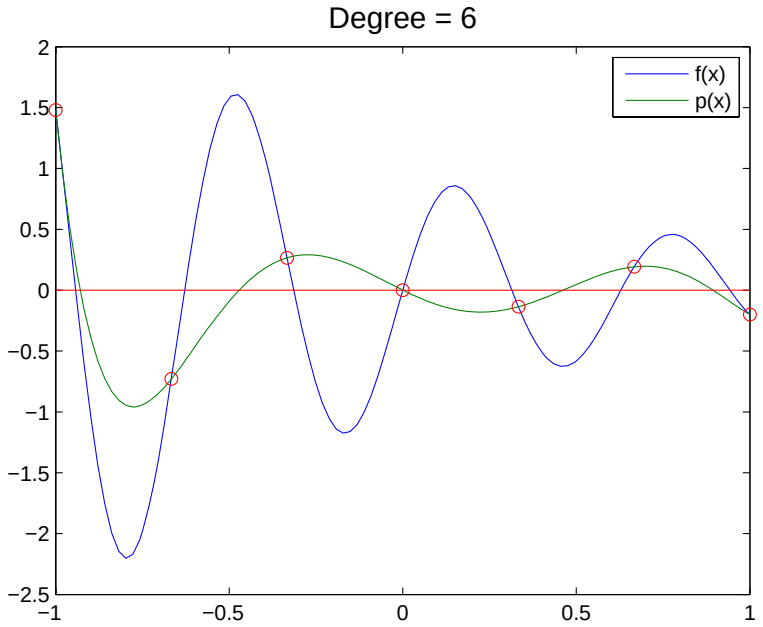
then $p_3(x_i) = y_i$, $i = 1{:}4$. We say that $p_3$ interpolates the data.

For a given $n$ let $p_n(x)$ be the polynomial interpolant of the function

$$f(x) \;=\; e^{-x}\sin(10x)$$

at the points defined by `linspace(-1,1,n+1)`.

Write a function `P2(n)` that plots both $f$ and $p_n$ across the interval $[-1, 1]$. Use \ to solve the underlying linear system. Sample output for the case $n = 6$:




Submit a listing of `P2` and output from the example $n = 12$.

*Review.* `plot`, `axis`, `title`, `legend`, `for`, `while`, `linspace`, `ones`, `zeros`, `subfunctions`, `length`, `sprintf`.

### P3. (A Fast Matrix-Vector Product)

Recursively define the $2^k$-by-$2^k$ matrix $W_k$ by

$$W_k \;=\; \frac{1}{2}\begin{bmatrix} W_{k-1} & W_{k-1} \\ W_{k-1} & -W_{k-1} \end{bmatrix}$$

with $W_0 = [1]$. Suppose $x \in \mathbb{R}^n$ where $n = 2^k$ and set $m = n/2$. If $x_T = x(1{:}m)$ and $x_B = x(m+1{:}n)$, then

$$y \;=\; W_k x \;=\; \frac{1}{2}\begin{bmatrix} W_{k-1} & W_{k-1} \\ W_{k-1} & -W_{k-1} \end{bmatrix}\begin{bmatrix} x_T \\ x_B \end{bmatrix} \;=\; \frac{1}{2}\begin{bmatrix} z_T + z_B \\ z_T - z_B \end{bmatrix}$$

with $z_T = W_{k-1}x_T$ and $z_B = W_{k-1}x_B$. Thus, a $W_k$ product reduces to a pair of $W_{k-1}$ products–a typical divide-and-conquer set-up. Develop a recursive implementation of the following function:

```
function Y = FastProd(X)
% X is 2^k -by- m for some k>=0.
% Y = W_k * X.
```

Explore the efficiency of `FastProd` by running the test script `P3`. Submit output and a listing of `FastProd`.

*Review.* `tic`, `toc`, `fprintf`.

**P4. (Structured Matrix-Vector Products)**

Complete the following function so that it performs as specified

```
function v = FirstCol(H,alfa)
% H is an n-by-n matrix with the property that H(i,j) = 0 whenever i>j+1.
% alfa is a column p-vector with p<n
% v is the first column of the matrix (H-alfa(1)*I)*(H - alfa(2)*I)* ... *(H - alfa(p)*I)
```

Your implementation should be vectorized, column-oriented, and flop-efficient. Submit a listing of `FirstCol` together with the output produced when the test script P4 is run.

*Review.* `eye`, `format`