HW 3

1: Tree climbing Let p be a *parent vector* for a tree with nodes $1, \ldots, n$, i.e. p_i is the index of the parent of node i (or zero if i is the root). Assume that $p_i < i$, so that children appear after parents in the ordering. Now, suppose that A is an $n \times n$ matrix such that

$$A_{ij} = \begin{cases} \alpha_i, & i = j, \\ \beta_j, & i = p_j, \\ \beta_i, & j = p_i, \\ 0, & \text{otherwise}, \end{cases}$$

and that A is positive definite. Write a function (not using the sparse intrinsics in MATLAB) that solves linear systems of the form $Ax = b$ in $O(n)$ time. Your function should look like

```
function [x] = p1tree(p, alpha, beta, b);
% Equivalent to
%   A = diag(alpha);
%   for k = 2:n
%     A(k,p(k)) = beta(k);
%     A(p(k),k) = beta(k);
%   end
%   x = A\b;
```

2: Going angling Suppose $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{n \times p}$. Write a routine to find x and y such that $|\cos(\theta(Ax, By))| = |(Ax) \cdot (By)|/(\|Ax\|\|By\|)$ is maximal — that is, render the acute angle between $\pm Ax$ and By as small as possible. Note: it may be helpful to read Section 2.6. Your routine should have the form

```
function [x,y] = p2angle(A,B)
% Find x and y to minimize the angle between Ax and By
```

Hint: For testing, it may be helpful to note that the best angle should be zero (*i.e.* $\cos(\theta) = 1$) when $m + p > n$.

3: Constrained optimization Consider the problem

$$\text{minimize} \|Ax - b\|_2 \text{ subject to } Bx = 0,$$

where $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times n}$ respectively have full row and column rank, and where $n + p < m$. The following MATLAB algorithm solves this problem:

```
function [x,mu] = constrained_ls(A,b,B)
% Minimize norm(Ax-b) subject to Bx = 0.

[Q,R] = qr(A,0);  % The "economy" QR
C  = B/R;         % C = B*R^{-1}
d  = Q'*b;        % Project b onto range of A
mu = C'\d;        % Minimize norm(C'*mu-d)
r  = d-C'*mu;     % Residual of minimization
x  = R\r;         % Compute constrained minimizer
```

Explain why this routine computes the solution as claimed (it may be helpful to first read section 12.1, and particularly 12.1.4).

4: Something sparse Modify the routine of question 3 so that it will be efficient if A is large and sparse, B is dense, and p is small. For this task, you should use one of the "Q-less" sparse QR routines, preferably with a fill-reducing ordering; see the MATLAB help for `qr` for details. Your modified routine should be named `p4solve`.