Assignment 4 - Solutions

Due: October 23, in class
No late assignments accepted

Issued: October 16, 2013

Important:

- Give complete answers: Do not only give mathematical formulae, but explain what you are doing. Conversely, do not leave out critical intermediate steps in mathematical derivations.
- Write your **name** as well as your **Sunet ID** on your assignment. **Please staple pages together.**
- Questions preceded by $\star$ are harder and/or more involved.
- **Include code with your assignment.**
- Comment any graphs and plots on the same page as the graph or plot itself.

Problem 1

In assignment 2, we discretized the 1-dimensional heat equation with Dirichlet boundary conditions:

$$\frac{d^2T}{dx^2} = 0, 0 \leq x \leq 1$$
$$T(0) = 0, T(1) = 2$$

The discretization leads to the matrix-vector equation $At = b$, with

$$A = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & -2 & 1 \\ 0 & \dots & 0 & 1 & -2 \end{pmatrix}, b = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -2 \end{pmatrix}$$

Here A is an $(N-1) \times (N-1)$ matrix.

a. (10 pts) Find the LU factorization of A for $N = 10$ using `Matlab`. Is `Matlab` using any pivoting to find the LU decomposition? Find the inverse of A also. As you can see the inverse of A is a dense matrix. **Note:** The attractive sparsity of A has been lost when computing its inverse, but L and U are sparse. Generally speaking, banded matrices have L and U with similar band structure. Naturally we then prefer to use the L and U matrices to compute the solution, and not the inverse. Finding L and U matrices with least "fill-in" ("fill-in" refers to nonzeros appearing at locations in the matrix where A has a zero element) is an active research area, and generally involves sophisticated matrix re-ordering algorithms.

Solution:

The solution can be found using Matlab command `lu`. To check whether Matlab is using any pivoting we can see what permutation matrix P is returned by the `lu` command. We can see it if we use `[L, U, P] = lu(A)` as in the code given below. For the above matrix, with $N = 10$ we get L, U, P as follows:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1/2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2/3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3/4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4/5 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -5/6 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -6/7 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -7/8 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -8/9 & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -3/2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4/3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5/4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -6/5 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -7/6 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -8/7 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -9/8 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10/9 \end{pmatrix}$$

and P is the identity matrix. Hence we know that `Matlab` is not using any pivoting in this case. While L and U maintain the sparsity of A, the A^{-1} does not.

$$A^{-1} = \begin{pmatrix} 0.9 & 0.8 & 0.7 & 0.6 & 0.5 & 0.4 & 0.3 & 0.2 & 0.1 \\ 0.8 & 1.6 & 1.4 & 1.2 & 1.0 & 0.8 & 0.6 & 0.4 & 0.2 \\ 0.7 & 1.4 & 2.1 & 1.8 & 1.5 & 1.2 & 0.9 & 0.6 & 0.3 \\ 0.6 & 1.2 & 1.8 & 2.4 & 2.0 & 1.6 & 1.2 & 0.8 & 0.4 \\ 0.5 & 1.0 & 1.5 & 2.0 & 2.5 & 2.0 & 1.5 & 1.0 & 0.5 \\ 0.4 & 0.8 & 1.2 & 1.6 & 2.0 & 2.4 & 1.8 & 1.2 & 0.6 \\ 0.3 & 0.6 & 0.9 & 1.2 & 1.5 & 1.8 & 2.1 & 1.4 & 0.7 \\ 0.2 & 0.4 & 0.6 & 0.8 & 1.0 & 1.2 & 1.4 & 1.6 & 0.8 \\ 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 \end{pmatrix}$$

Matlab code for solving part a)

```
% Assignment 4 - problem a , finding LU and inverse of A of a
% finite difference matrix to solve T_xx = f(x), 0<=x<=1
% with T( x=0) = 0 , T( x=1) = 2
clear all
N = 10;

% Find the points of discretization
h = 1/N;
% Interval size of discretization
x = 0:h:1; % x = [x_0, x_1, ..., x_N]
```

```
x = x(2:end-1); % x = [x_1, x_2, ..., x_{N-1}]
% Construct the tri-diagonal matrix A
A = diag(ones(N-2,1),1) - 2*eye(N-1) + diag(ones(N-2,1),-1);

% Instead of the line above, use the line below to take advantage of the sparsity of A.
% We can construct A as a sparse matrix directly (this will save us some memory space
% and speed up computation)
% A = spdiags(ones(N-1,2), [-1;1], N-1, N-1) - 2*speye(N-1);
% Note: to answer this question, we don't need to setup the RHS, b

[L, U, P ] = lu(A);
sym(L)
sym(U)
A_inv = inv(A)
```

b. (10 pts) Compute the determinants of L and U for $N = 1000$ using `Matlab`'s determinant command. Why does the fact that they are both nonzero imply that A is non-singular? How could you have computed these determinants really quickly yourself without `Matlab`'s determinant command?

Solution:

Determinants can be computed using Matlab command `det`. The determinant of L is 1 and that of U is -1000. The same answer could have been obtained by hand by simply multiplying the diagonal elements of the triangular matrices.

The fact that the determinants of L and U are both nonzero implies that A is nonsingular since,

$|A| = |L||U| = 0$

Problem 2

a. (i) (10 pts) Use `Matlab` command `A=rand(4)` to generate a random 4-by-4 matrix and then use the function `qr` to find an orthogonal matrix Q and an upper triangular matrix R such that $A = QR$. Compute the determinants of A, Q and R.

Solution:

```
A = rand(4);
[Q, R]= qr(A);
det(A)
det(Q)
det(R)
```

(ii) (15 pts) Set `A=rand(n)` for at least 5 different n's in `Matlab` for computing the determinant of Q where Q is the orthogonal matrix generated by `qr(A)`. What do you observe about the determinants of the matrices Q? Show, with a mathematical proof, that the determinant of any orthogonal matrix is either 1 or -1.

Solution:

$Q^T Q = I \Rightarrow |QTQ| = 1 \Rightarrow |Q|^2 = 1$

(iii) (15 pts) For a square $n \times n$ matrix matrix B, suppose there is an orthogonal matrix Q and an upper-triangular matrix R such that $B = QR$. Show that if a vector x is a linear combination of the first k column vectors of B with $k \leq n$, then it can also be expressed as a linear combination of the first k columns of Q.

Solution:

Let $\vec{b_i}$, $\vec{q_i}$ be the i-th column of B and Q respectively. Then by expanding QR, we have

$$\vec{b_k} = \sum_{i=1}^{n} R_{ik}\vec{q_i} = \sum_{i=1}^{k} R_{ik}\vec{q_i}$$

The last equal sign holds because R is an upper triangular matrix and so $R_{ik} = 0$ for all $i > k$. Therefore, $\vec{b_j} \in \text{span}\{\vec{q_1}, \vec{q_2}, \ldots, \vec{q_k}\}$ for all $j \leq k$. And so, $x \in \text{span}\{\vec{b_1}, \vec{b_2}, \ldots, \vec{b_k}\} \Rightarrow x \in \text{span}\{\vec{q_1}, \vec{q_2}, \ldots, \vec{q_k}\}$.

b*. (i) (5 pts) Assume $\{\vec{v_1}, \vec{v_2} \cdots \vec{v_n}\}$ is an orthonormal basis of $\mathbb{R}^n$. Suppose there exists a unit vector $\vec{u}$ such that $\vec{u}^T\vec{v_k} = 0$ for all $k = 2, 3 \cdots n$, show that $\vec{u} = \vec{v_1}$ or $\vec{u} = -\vec{v_1}$.

Solution: Let V be the matrix with $\vec{v_k}$ as columns. Then since the $\vec{v_k}$ are an orthonormal basis, we have $V^TV = I$ and $V\vec{\alpha} = \vec{u}$, for some vector $\vec{\alpha}$.

$$V^T\vec{u} = \gamma\vec{e_1} \Rightarrow V^TV\vec{\alpha} = \gamma\vec{e_1}$$

Where $\vec{e_1}$ is the first elementary vector. Using the second equality, we have $\vec{\alpha} = \gamma\vec{e_1}$ and since $\vec{u}$ is a unit vector we must have $\gamma = 1$ or $\gamma = -1$. This implies $\vec{u} = \pm\vec{v_1}$

(ii) (5 pts) Prove that if $C = QR$, where Q is an orthogonal matrix and R is an upper-triangular matrix with diagonal elements all positive, then the Q and R are unique.

Solution: We want to show if there are any orthogonal matrix P and upper triangular matrix S with positive diagonal elements such that $PS = C$, then $Q = P$ and $R = S$. Consider,

$$\begin{aligned}
QR &= PS \\
\Rightarrow P^TQ &= SR^{-1} \text{ , an arbitrary upper triangular matrix} \\
\Rightarrow \vec{p_i}^T\vec{q_j} &= 0 \;\text{ for all } i < j \\
\text{Similarly...} & \\
PS &= QR \\
\Rightarrow Q^TP &= RS^{-1} \text{ , an arbitrary upper triangular matrix} \\
\Rightarrow \vec{q_i}^T\vec{p_j} &= 0 \;\text{ for all } i < j
\end{aligned}$$

So we have $\vec{p_i}^T\vec{q_j} = 0$ for all $i \neq j$ and $\vec{p_i}$ is a unit vector. So by the previous question, $\vec{p_i} = \pm\vec{q_i}$. Finally since $R_{ii} > 0, S_{ii} > 0$ for all i, and $(RS^{-1})_{ii} = \frac{R_{ii}}{S_{ii}} > 0 \Rightarrow \vec{p_i} = \vec{q_i}$ which shows $P = Q$ and $R = S$.

Another way to see this let $M = Q^TP = RS^{-1}$ implies that M is both orthogonal (since it is a product of orthogonal matrices) and upper triangular (since it is a product of upper triangular matrices). We therefore have that M must be the identity. That is, $\vec{m}_1 = \vec{e_1} \Rightarrow (\vec{m}_k)_1 = 0$ for all $k > 1$. Therefore, $\vec{m}_2 = \vec{e_2}$ which implies $(\vec{m}_k)_2 = 0$ for all $k > 2$ and so on which shows $M = I$.

Problem 3

In class we have introduced the LU decomposition of A, where L is unit-lower triangular, in that it has ones along the diagonal, and U is upper triangular. However, in the case of symmetric matrices, such as the discretization matrix, it is possible to decompose A as LDL^T, where L is still unit-lower triangular and D is diagonal. This decomposition clearly shows off the symmetric nature of A.

a. (10 pts) Find the LDL^T decomposition for the matrix given in **Problem 1**. Show that L is bidiagonal. How do D and L relate to the matrix U in the LU decomposition of A?
Hint: Think about how D and L^T relate to U.
Note: Computing LDL^T this way does not work out for any symmetric matrix, it only happens to work for this matrix in particular.

Solution: If we set $U = DL^T$, and if we keep in mind that L^T is required to be unit-upper triangular (ones on the diagonal), then we can simply factor out the diagonal elements of U as D and what's left over will be L^T. In other words

$$U = DL^T$$
$$L^T = D^{-1}U$$

D can be obtained by the `Matlab` command `D = diag(diag(U))` and by doing `Lt = diag(1./diag(D))*U`. It's interesting that for this particular matrix it turns out that the result is nothing more than the transpose of the L that was obtained by the `lu` command.

b*. (i) (10 pts) To solve $A\vec{x} = \vec{b}$ we can exploit this new decomposition. We get $LDL^T\vec{x} = \vec{b}$ which we can now break into three parts: Solve $L\vec{y} = \vec{b}$ using forward substitution, now solve $D\vec{z} = \vec{y}$, and then solve $L^T\vec{x} = \vec{z}$ using back substitution. Write a `Matlab` code that does exactly this for arbitrary N for the A in **Problem 1**.

Solution: The idea here is that we solve the system in a series of steps

$$Ax = b$$
$$LDL^T x = b$$

First solve this system using forward substitution (taking note that each equation in this system has at most two variables since L is bi-diagonal)

$$Lz = b$$

Then solve this system by simply dividing by the diagonal element of D

$$Dy = z$$

Finally solve this system by backward substitution

$$L^T x = y$$

Note that your code does not need to compute L^T the same way as part a, you may directly use the transpose of the L obtained from `lu`. You are also allowed to use `ldl` to get L and D.

(ii) (5 pts) Solve a system of the same form as **Problem 1** for A of size 10 and of size 1000 with $\vec{b}$ having all zeros except 2 as the last entry in both cases, and verify the correctness of your solution using `Matlab`'s `A\b` operator and the `norm` command.

Solution:

```
N = size(A, 1);
b = zeros(N, 1);
b(N) = 2;

[L, U, P] = lu(A);
```

```
D = diag(diag(U));

% solve Ly=b
y = zeros(N, 1);
y(1) = b(1);
for i=2:N
    y(i) = b(i) - L(i, i-1) * y(i-1);
end

% solve Dz = y
z = y .* (1 ./ diag(D));

% solve L^Tx = z
Lt = L'; % for convenience
x = zeros(N, 1);
x(N) = z(N);
for i=(N-1):-1:1
  x(i) = z(i) - Lt(i, i+1) * x(i+1);
end

norm(A \ b - x)
```

for both $N = 10$ and $N = 1000$ the norm difference is zero.