Spring 2018 Math 541

Homework – Numerical Linear Algebra solutions Due Tues. 3/9/18

Be sure to include all MatLab programs used to obtain answers.

1. (10pts) (From the text, **2.6**.) This question is

2.6. This example shows that a badly conditioned matrix does not necessarily lead to small pivots in Gaussian elimination. The matrix is the $n \times n$ upper triangular matrix A with elements

$$a_{ij} = \begin{cases} -1, & i < j. \\ 1, & i = j \\ 0, & i > j \end{cases}$$

We want to show this matrix is ill-conditioned by showing that

$$\kappa_1(A) = n2^{n-1}.$$

We also want to find for what n does $\kappa_1(A)$ exceed 1/eps?

Solution:
To generate this matrix in MatLab with eye, ones, and triu. Then set $n =$ some number and write the MatLab command

```
eye(n)- triu(ones(n),1)
```

So, as always, it helps to visualize the matrix A you are dealing with which looks like

$$A = \begin{pmatrix} 1 & -1 & -1 & \cdots & -1 \\ 0 & 1 & -1 & \cdots & -1 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -1 \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$$

Recall that the condition number $\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1$, so we need to find the norm of A and its inverse.

In the 1-norm, we have that

$$\|A\vec{x}\|_1 = \sum_{j=1}^{n-1} \left| x_j - \sum_{k=j+1}^{n} x_k \right| + |x_n|$$

As for establishing the bound on A, using the triangle equality, we have that

$$\sum_{j=1}^{n-1} \left| x_j - \sum_{k=j+1}^{n} x_k \right| + |x_n| \leq \|\vec{x}\|_1 + \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} |x_k| \leq \|\vec{x}\|_1 + (n-1)\|\vec{x}\|_1 = n\|\vec{x}\|_1,$$

since

$$\sum_{k=j+1}^{n} |x_k| \leq \|\vec{x}\|_1.$$

Using the vector

$$\vec{e}_n = [0 \ \cdots \ 0 \ 1]^T,$$

in place of $\vec{x}$, we readily see that

$$\|A\vec{e}_n\|_1 = n, \ \|\vec{e}_n\| = 1,$$

and thus we have that

$$\|A\|_1 = n.$$

As for the inverse, if we look at the 3×3 case, we see that

$$A^{-1} = \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

and the 4×4 case is given by

$$A^{-1} = \begin{pmatrix} 1 & 1 & 2 & 4 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

We thus see that one inverse is nested within the other. But this makes good sense since I can write the $n+1$ dimensional matrix A, which we label as A_{n+1} as

$$A_{n+1} = \left(\begin{array}{c|c} A_n & -\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \\ \hline 0 \cdots 0 & 1 \end{array} \right),$$

then we must have

$$A_{n+1}^{-1} = \left(\begin{array}{c|c} A_n^{-1} & \vec{c} \\ \hline 0 \cdots 0 & 1 \end{array} \right).$$

Doing the matrix multiplication then gives us

$$\vec{c} = A_n^{-1} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

At this point, we make the inductive hypothesis that

$$A_n^{-1} = \begin{pmatrix} 1 & 1 & 2 & \cdots & 2^{n-2} \\ 0 & 1 & 1 & \cdots & 2^{n-3} \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & \cdots & 1 & 1 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix},$$

so that using basic geometric series arguments, we see that

$$\vec{c} = \begin{pmatrix} 2^{n-1} \\ 2^{n-2} \\ \vdots \\ 2 \\ 1 \end{pmatrix}.$$

Thus, we see the inductive hypothesis is correct, and we have found the form of the inverse.

In the 1-norm, we have that

$$\|A^{-1}\vec{x}\|_1 = \sum_{j=1}^{n-1} \left| x_j - \sum_{k=j+1}^{n} 2^{k-(j+1)} x_k \right| + |x_n|$$

Using the triangle equality and rearranging the terms, we have that

$$\|A^{-1}\vec{x}\|_1 \le 2^{n-1}\|\vec{x}\|_1.$$

Again with $\vec{e}_n$, we see that

$$\|A^{-1}\|_1 = \frac{\|A^{-1}\vec{x}\|_1}{\|\vec{x}\|_1} = 2^{n-1}.$$

It follows that the condition number is

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 = n2^{n-1}.$$

Thus, looking at

$$n2^{n-1} = 1/eps,$$

or

$$\log_2(n) + n - 1 = -\log_2(eps),$$

using Newton's method, we can readily find that

$$n \approx 47.4,$$

so rounding up, for $n \ge 48$, the condition number is larger than the reciprocal of machine precision. This implies that we should be very cautious about working with A if the dimension is larger than this value since we anticipate roundoff error completely dominating any results.

As for finding an $\vec{x}$ such that $||A\vec{x}||_1 \ll ||\vec{x}||_1$, I would try

$$\vec{x} = \begin{pmatrix} 1 \\ 1/2 \\ 1/4 \\ \vdots \\ 2^{-n+1} \end{pmatrix}.$$

In this case,

$$||\vec{x}||_1 = 2 - 2^{-n+1}.$$

We then can show

$$||A\vec{x}||_1 = n2^{-n+1}.$$

Thus in the limit of large n, $||A\vec{x}||_1 \ll ||x||_1$. This example shows how one might see the effects of the ill-conditioning of this matrix since one could get, due to roundoff, $A\vec{x} \approx 0$ for $n \geq 48$.

Clearly the pivots with respect to Gaussian elimination are all one.

2. (5 pts) (From the text, **2.7**.) The matrix factorization

$$LU = PA$$

can be used to compute the determinant of A... modify the lutx function so that it returns four outputs.

Provide two test cases, which show your approach is correct. Note that a good way to produce random $N \times N$ matrices in MatLab is to use the command `randn(N)`. Use this for examples in this problem and the next.

Solution: The magnitude of the determinant is easily found by multiplying the diagonal elements of the Upper Triangular matrix U. The only part missing in the existing program is finding the sign of $\det(A)$. The `lutx.m` file needs to be modified to determine the number of pivots that occur, which will swap the sign of the determinant for each pivot. If the number of pivots is odd, then we obtain $-\det(U)$, while if it is even then $\det(A) = \det(U)$. One easy modification to `lutx` is the following:

```
function [L,U,p,sig] = lutx(A)
[n,n] = size(A);
p = (1:n)';
sig = 1;
for k = 1:n-1
    [r,m] = max(abs(A(k:n,k)));
    m = m+k-1;
    if (A(m,k) ≠ 0)
        if (m ≠ k)
        A([k m],:) = A([m k],:);
        p([k m]) = p([m k]);
        sig = -sig;
        end
    A(k+1:n,k) = A(k+1:n,k)/A(k,k);
    A(i,k+1:n) = A(i,k+1:n) - A(i,k)*A(k,k+1:n);
    end
end
L = tril(A,-1) + eye(n,n);
U = triu(A);
```

Once we have this, in order to find a determinant, we need only find `sig*prod(diag(U))`. Using this approach, we look at the test cases

```
A = randn(10);
det(A) = 3.237446354875954e+03
[L,U,p,sig] = lutx(A);
sig*prod(diag(U)) = 3.237446354875954e+03
```

and

```
A = randn(100);
det(A) =  -1.366073030472072e+76
[L,U,p,sig] = lutx(A);
sig*prod(diag(U)) =  -1.366073030472070e+76
```

We see that our approach is clearly accurate across a wide range of dimensions.

3. (12 pts) In class we solved $Ax = b$ using a **Direct Method**, specifically Gaussian elimination with partial pivoting. This used a fair number of steps, which went up computationally like n^3, where $n \times n$ was the dimension of the matrix A. Often large matrices, especially **sparse** matrices are solved iteratively. These iterative methods are particularly valuable when the matrix is **diagonal dominant**, which is where the value of the diagonal elements of A exceed the sum of the other elements in the row.

a. Suppose the matrix A is written:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

and split this matrix into a diagonal matrix D, a lower triangular matrix L, and an upper triangular matrix U. Write the matrix:

$$\begin{aligned} A &= \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix} - \begin{pmatrix} 0 & 0 & \cdots & 0 \\ -a_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{pmatrix} - \begin{pmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -a_{n-1,n} \\ 0 & \cdots & 0 & 0 \end{pmatrix} \\ &= D - L - U \end{aligned}$$

Show that if D^{-1} exists ($a_{ii} \neq 0$ for $1 \leq i \leq n$), then the equation

$$Ax = (D - L - U)x = b$$

has the solution

$$x = D^{-1}(L + U)x + D^{-1}b.$$

Solution:

$$\begin{aligned} Dx - Lx - Ux &= b \\ Dx &= (L + U)x + b \\ D^{-1}Dx &= D^{-1}(L + U)x + D^{-1}b \\ x &= D^{-1}(L + U)x + D^{-1}b \end{aligned}$$

b. If $T = D^{-1}(L + U)$ and $c = D^{-1}b$, then **Jacobi's iterative method** satisfies the formula

$$x^{(k)} = D^{-1}(L + U)x^{(k-1)} + D^{-1}b = Tx^{(k-1)} + c.$$

Consider the system:

$$\begin{pmatrix} 4 & 1 & -1 & 1 \\ 1 & 4 & -1 & -1 \\ -1 & -1 & 5 & 1 \\ 1 & -1 & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \\ 0 \\ 1 \end{pmatrix}$$

Find T and c. Let $x^{(0)} = \mathbf{0}$, the zero vector, and find the first **3** iterations using Jacobi's iterative method.

Solution:

$$D = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix} \qquad L = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ -1 & 1 & -1 & 0 \end{pmatrix} \qquad U = \begin{pmatrix} 0 & -1 & 1 & -1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Then

$$D^{-1} = \begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{5} & 0 \\ 0 & 0 & 0 & \frac{1}{3} \end{pmatrix} \qquad (L+U) = \begin{pmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & 1 & 1 \\ 1 & 1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{pmatrix}$$

Then

$$T = D^{-1}(L+U) = \begin{pmatrix} 0 & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{5} & \frac{1}{5} & 0 & -\frac{1}{5} \\ -\frac{1}{3} & \frac{1}{3} & -\frac{1}{3} & 0 \end{pmatrix} \qquad c = D^{-1}b = \begin{pmatrix} -\frac{1}{2} \\ -\frac{1}{4} \\ 0 \\ \frac{1}{3} \end{pmatrix}$$

The first three iterations are then

$$x^{(1)} = \begin{pmatrix} -0.500000000000000 \\ -0.250000000000000 \\ 0.00000000000000 \\ 0.333333333333333 \end{pmatrix} \quad x^{(2)} = \begin{pmatrix} -0.520833333333333 \\ -0.041666666666667 \\ -0.216666666666667 \\ 0.416666666666667 \end{pmatrix} \quad x^{(3)} = \begin{pmatrix} -0.647916666666667 \\ -0.069791666666667 \\ -0.195833333333333 \\ 0.565277777777778 \end{pmatrix}$$

c. Continue Jacobi's iterative method for **10** iterations. Give the value of $x^{(10)}$ Also, find the exact solution, x^*, using Matlab's `linsolve` program. Determine the error between the 10th iterate and the exact solution in the 1-norm, 2-norm, and ∞-norm, *i.e.*, find

$$\|x^{(10)} - x^*\|_1 \quad \text{and} \quad \|x^{(10)} - x^*\|_2 \quad \text{and} \quad \|x^{(10)} - x^*\|_\infty .$$

Using

```
function xout = jacobihw(n)
%using the Jacobian from question 4
%   homework 5
T = [0 -1/4 1/4 -1/4 ; -1/4 0 1/4 1/4; 1/5 1/5 0 -1/5; -1/3 1/3 -1/3 0];
c = [-1/2;-1/4;0;1/3];
```

```

i = 0;
x0=[0; 0;0;0];
i=1;
for i=1:n
    x0 = T*x0+c;
    i=i+1

end
    format long;
    xout =x0;

end
```

the 10th iteration gives

$$x = \begin{pmatrix} -0.749656471691995 \\ 0.038916625662596 \\ -0.279350155049391 \\ 0.687092605276853 \end{pmatrix} \qquad \text{linsolve} \quad x^* = \begin{pmatrix} -0.753424657534247 \\ 0.041095890410959 \\ -0.280821917808219 \\ 0.691780821917808 \end{pmatrix}$$

And by changing this program slightly to include the `linsolve` we can find the 1-norm, 2-norm, and ∞-norm,

$$\|x^{(10)} - x^*\|_1 = 0.012107429990398 \quad \text{and}$$

$$\|x^{(10)} - x^*\|_2 = 0.006564592925384 \quad \text{and}$$

$$\|x^{(10)} - x^*\|_\infty = 0.004688216640955.$$

```
  A=[4 1 -1 1 ;1 4 -1 -1;-1 -1 5 1; 1 -1 1 3];
  b = [-2; -1; 0; 1];
  xs =linsolve(A,b);
   xs= x0-xs;
   x1=norm(xs,1)
   x2=norm(xs,2)
   xinf=norm(xs, inf)
end
```