MATH-6800
Vignesh Ramakrishnan
RIN: 662028006
Due: Thursday November 17, 2022

MATH 6800: Problem Set 7

1. A matrix D is block tridiagonal if it is of the form

$$D = \begin{bmatrix} B_1 & C_1 & & & & \\ A_2 & B_2 & C_2 & & & \\ & A_3 & B_3 & C_3 & & \\ & & A_4 & B_4 & C_4 & \\ & & & \ddots & \ddots & \ddots \\ & & & & A_n & B_n \end{bmatrix}$$

where each A_i, B_i and C_i is a small matrix of size $p \times p$ (p is the block size). Derive a block LU decomposition (i.e. an LU decomposition that uses operations involving $p \times p$ matrices instead of scalars), assuming no pivoting. What are the conditions you need for this LU decomposition to exist?

Assuming every block matrix (A_i, B_i, C_i) is nonsingular then let the operations be performed as:

$$\begin{bmatrix} I & & & \\ -A_2\,(B_1)^{-1} & I & & \\ & & I & \\ & & & \ddots \end{bmatrix} \begin{bmatrix} B_1 & C_1 & & & \\ A_2 & B_2 & C_2 & & \\ & A_3 & B_3 & C_3 & \\ & & \ddots & \ddots & \ddots \end{bmatrix} = \begin{bmatrix} B_1 & C_1 & & & & \\ & \left(-A_2\,(B_1)^{-1}\,C_1 + B_2\right) & C_2 & & & \\ & A_3 & B_3 & C_3 & & \\ & & A_4 & B_4 & C_4 & \\ & & & \ddots & \ddots & \ddots \end{bmatrix}$$

$$\begin{bmatrix} I & & & \\ & I & & \\ & \left(-A_3\left(-A_2\,(B_1)^{-1}\,C_1 + B_2\right)^{-1}\right) & I & \\ & & & \ddots \end{bmatrix} \begin{bmatrix} B_1 & C_1 & & & \\ & \left(-A_2\,(B_1)^{-1}\,C_1 + B_2\right) & C_2 & & \\ & A_3 & B_3 & C_3 & \\ & & \ddots & \ddots & \ddots \end{bmatrix} =$$

$$\begin{bmatrix} B_1 & C_1 & & \\ & \left(-A_2\,(B_1)^{-1}\,C_1 + B_2\right) & C_2 & \\ & & \left(-A_3\left(-A_2\,(B_1)^{-1}\,C_1 + B_2\right)^{-1} C_2 + B_3\right) & C_3 \\ & & & \ddots \end{bmatrix}$$

Therefore the algorithm can be written as:

$$\begin{aligned} \alpha_1 &= \mathbf{0}_{p\times p}, & \beta_1 &= B_1 \\ \alpha_i &= A_i\,(\beta_{i-1})^{-1} & \beta_i &= -\alpha_i C_{i-1} + B_i,\ \text{i} = 2,\,3,\,4,\ldots p \end{aligned}$$

In this case, we require every β_i matrix to be nonsingular.

$$\begin{bmatrix} I & & & & \\ \alpha_2 & I & & & \\ & \alpha_3 & I & & \\ & & & \ddots & \\ & & & \alpha_n & I \end{bmatrix} \begin{bmatrix} \beta_1 & C_1 & & & \\ & \beta_2 & C_2 & & \\ & & \beta_3 & C_3 & \\ & & & \ddots & \\ & & & & \beta_n \end{bmatrix} = LU$$

2. NLA 20.3 *Suppose an $m \times m$ matrix $A \ldots$*

The matrix $A \in \mathbb{C}^{m \times m}$ is written as a block matrix with $A_{11}, A_{12}, A_{21}, A_{22}$ where $A_{11} \in \mathbb{C}^{n \times n}$ and $A_{22} \in \mathbb{C}^{m-n \times m-n}$.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

(a)

$$\begin{bmatrix} I & \mathbf{0} \\ -A_{21}\left(A_{11}\right)^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ -\cancel{A_{21}}\left(A_{11}\right)^{-1} A_{11} + \cancel{A_{21}} & A_{21}\left(A_{11}\right)^{-1} A_{12} + A_{22} \end{bmatrix}$$
$$= \begin{bmatrix} A_{11} & A_{12} \\ \mathbf{0} & A_{22} - A_{21}\left(A_{11}\right)^{-1} A_{12} \end{bmatrix}$$

(b) $A_{21} \in \mathbb{C}^{m-n \times n}$. Since there are n columns, there are n Gauss Elimination steps. Show that this still results in $A_{22} - A_{21}\left(A_{11}\right)^{-1} A_{12}$ in the bottom right.

3. The matrix $A \in \mathbb{C}^{m \times m}$ is diagonally dominant if

$$|a_{ii}| > \sum_{j=1, j \neq i}^{m} |a_{ij}|, \;\; i = 1, 2, \ldots, m.$$

(a) Prove that if A is diagonally dominant, then any principle submatrix of A is diagonally dominant.

To generate any principle submatrix, if k rows are removed, then the same k columns are to be removed. So, if $2, 3, 5$ rows are removed, then $2, 3, 5$ columns are also removed. In which case, the resulting matrix is also a square matrix of the form $\tilde{A} \in \mathbb{C}^{m-3 \times m-3}$. When generalized, this resulting square matrix is $\tilde{A} \in \mathbb{C}^{m-k \times m-k}$. Now in this particular example, if we take any row $r \neq 2, 3, 5$ from A matrix,

$$\sum_{j=1, j \neq 2,3,5}^{m} |a_{rj}| = |a_{r1}| + |a_{r2}| + \ldots + |a_{rr}| + \ldots + |a_{rm}| - |a_{r2}| - |a_{r3}| - |a_{r5}|$$

If $|a_{rr}| > \sum_{j=1, j \neq r}^{m} |a_{rj}|$ then it is definitely greater than, $\left(\sum_{j=1, j \neq r}^{m} |a_{rj}|\right) - |a_{r2}| - |a_{r3}| - |a_{r5}|$ since those are absolute values which are just positive values being subtracted. Therefore, if k rows and k columns are removed, then k positive values are subtracted from the summation. Hence, any principle submatrix is diagonally dominant.

(b) Prove that if A is diagonally dominant, then A is nonsingular.

If a matrix is singular, then there exists a vector $u \in \mathbb{C}^m \backslash \{0\}$ such that $Au = \mathbf{0}$. In which case, we have

$$\sum_{j=1}^{m} a_{ij} u_j = 0$$

In the worst case scenario, if for any row k if $u_k = ||u||_\infty$,

$$\begin{aligned} \sum_{j=1}^{m} a_{kj} \frac{u_j}{u_k} &= 0 \\ -\sum_{j=1, j\neq k}^{m} a_{kj} \frac{u_j}{u_k} &= a_{kk} \\ \left| \sum_{j=1, j\neq k}^{m} a_{kj} \frac{u_j}{u_k} \right| &= |a_{kk}| \end{aligned}$$

Then, we have

$$|a_{kk}| \leq \sum_{j=1, j\neq k}^{m} |a_{kj}| \left| \frac{u_j}{u_k} \right|$$

This is a contradiction because $\left|\frac{u_j}{u_k}\right| \leq 1$ but $|a_{kk}| > \sum_{j=1, j\neq k}^{m} |a_{kj}|$. Therefore, A is nonsingular.

(c) Prove that if A is diagonally dominant then it will have an LU decomposition (you may use the result of NLA 20.1).

In NLA 20.1 we have proved that when A is nonsingular, a LU decomposition exists if $A_{1:k,1:k}$ is nonsingular. From part (a), we proved that every principle submatrix is diagonally dominant. In part (b) we proved that any diagonally dominant matrix is nonsingular. Hence any principle submatrix $A_{1:k,1:k}$ will be nonsingular since they are all diagonally dominant. So, with NLA 20.1, we can prove that A has a LU decomposition.

4. Write a Matlab code $[L, U, P] = lufactor(A)$ that takes an $m \times m$ matrix A and computes the LU factorization, $PA = LU$, using partial pivoting. Write a second Matlab code $x = lusolve(b, L, U, P)$ that solves the system $Ax = b$, for some x given b, using the output from $lufactor$. For this exercise you should only use elementary arithmetic, vector and matrix operations (e.g. no backlash operators to solve the triangular systems).

(a) Test your $lufactor(A)$ routine on this matrix,

$$A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix}$$

Output A, L, U and P. Check that $PA = LU$.

The code to compute L, U and P from A is in Listing 1.

```
function [L,U,P] = lufactor(A)

[m,~] = size(A);
U = A;
L = eye(m);
P = eye(m);

for k=1:m-1
    % finding maximum value in the column
    max_val = abs(U(k,k));
    max_id  = k;
    for i=k:m
        if(abs(U(i,k)) > max_val)
            max_val = abs(U(i,k));
            max_id  = i;
        end
    end

    % interchanging two rows
    t = U(k,k:m);
    U(k,k:m) = U(max_id,k:m);
    U(max_id,k:m) = t;

    t = L(k,1:k-1);
    L(k,1:k-1) = L(max_id,1:k-1);
    L(max_id,1:k-1) = t;

    t = P(k,:);
    P(k,:) = P(max_id,:);
    P(max_id,:) = t;

    for j=k+1:m
        L(j,k) = U(j,k)/U(k,k);
        U(j,k:m) = U(j,k:m) - L(j,k)*U(k,k:m);
    end

end

end
```

Listing 1: **lufactor**(A)

(b) Test your function *lusolve* by solving $Ax = b$ where A is from part (a) and $b = [7, 23, 69, 79]^T$. Output x and check that $Ax = b$.

This is solved in two parts, where

$$Ax = b$$
$$PAx = Pb$$
$$LUx = Pb$$

$$Ly = z \qquad\qquad Ux = y$$

The code used to solve this system of equations is in Listing 2. Script in Listing 3 is used to verify for the given case in consideration and the solutions are attached below as well.

```
function x = lusolve(b,L,U,P)

[m,~] = size(L);

% part 1
z = P*b;
y = zeros(m,1);
for i=1:m
    s = z(i);
    for j=1:i
        if j~=i
            s = s - L(i,j)*y(j);
        end
    end
    y(i) = s;
end

% part 2
x = zeros(m,1);
for i=m:-1:1
    if i==m
        x(i) = y(i)/U(i,i);
    else
        rv = U(i,i+1:m);
        xv = x(i+1:m,1);
        x(i) = (y(i)-rv*xv)/U(i,i);
    end
end

end
```

Listing 2: **lusolve(b,L,U,P)**

```
clc
clear
%%
A = [2 1 1 0; 4 3 3 1; 8 7 9 5; 6 7 9 8];

[L,U,P] = lufactor(A);
fprintf('Norm(PA-LU): %f \n',norm(P*A-L*U));

b = [7 23 69 79]';
x = lusolve(b,L,U,P);
disp('x=');
disp(x);
disp('Ax = ');
disp(A*x);
fprintf('Norm(Ax-b): %f \n', norm(A*x-b));
```

Listing 3: script

```
Norm(PA-LU): 0.000000
x=
    1.0000
    2.0000
    3.0000
    4.0000

Ax =
    7.0000
   23.0000
   69.0000
   79.0000

Norm(Ax-b): 0.000000
>>
```