Assignment 3: The LU and Cholesky Decomposition

Gregory Smetana
ID 1917370
ACM 106a

December 12, 2013

1 Why not to explicitly evaluate the inverse (unless absolutely necessary)

One algorithm to solve the matrix equation

$$AX = B \tag{1}$$

with $A \in \mathbf{R}^{nxn}$ and $B \in \mathbf{R}^{nxn}$ is to find A^{-1} using Gaussian elimination with partial pivoting and compute the product $X = A^{-1}B$. The inverse is given by the right part of the augmented matrix $(A|I)$ after the left part has been reduced to the identity matrix by performing elementary row operations. The inverse A^{-1} of A is the solution to the matrix equation

$$AX = I \tag{2}$$

The algorithm has four steps:

1. The entry in the left column with the largest absolute value (pivot) is found.
2. The rows are interchanged so that the pivot is in the first row
3. The first row is divided by the pivot.
4. Elementary row operations are used to reduce the other entries in the first column to zero.

The steps are repeated on each submatrix of A until it is in row echelon form. Back substitution is then used to reduce the left part of the augmented matrix to the identity. After the inverse is known, the product $X = A^{-1}B$ may be computed.

A second algorithm factorizes $A = PLU$ using Gaussian elimination and then solves for each column of X by forward and back substitution. This is done by applying Algorithm 2.1 from Demmel:

1. Factorize A into $A = PLU$, where
   P = permutation matrix
   L = unit lower triangular matrix
   U = nonsingular upper triangular matrix
2. Solve $PLUx = b$ for LUx: $LUx = P^T b$
3. Solve $LUx = P^T B$ for Ux by forward substitution: : $Ux = L^{-1}(P^T b)$
4. Solve $Ux = L^{-1}(P^T B)$ for x by back substitution: $x = U^{-1}(L^{-1}P^T b)$

Therefore, a formula for A^{-1} may be expressed as:

$$A^{-1} = U^{-1}L^{-1}P^T \tag{3}$$

From class, we know that the LU factorization requires $2/3n^3$ operations. After this is done, the solution for each column needs 2 substitutions with n^2 operations each. Therefore, solving the general matrix equation $AX = B$ requires $2/3n^3 + 2n^2m$ operations. If $B = I$, the total number of operations required is approximately $8/3n^3$.

Solving the linear system $Ax = b$ by first computing the inverse therefore requires $\approx 8/3n^3$ operations. Solving the system by Gaussian elimination with partial pivoting requires $\approx 2/3n^3$ operations. Therefore, one should never explicitly evaluate the inverse unless absolutely necessary.

2 The LDL Decomposition

Suppose that $A \in \mathbf{R}^{nn}$ is nonsingular and there exists unique lower triangular matrix $L \in \mathbf{R}^{nn}$ with diagonal entries equal to 1 and unique upper triangular matrix $U \in \mathbf{R}^{nn}$ such that

$$A = LU \tag{4}$$

Since A is nonsingular, it follows that U is nonsingular. Since U is triangular, all of the diagonal entries of U must be non-zero. This means the diagonal matrix

$$D = \begin{pmatrix} u_{1,1} & & 0 \\ & \ddots & \\ 0 & & u_{n,n} \end{pmatrix} \tag{5}$$

is also nonsingular. Therefore, there exists a unique upper triangular matrix $\tilde{U} \in \mathbf{R}^{nxn}$ with all diagonal entries equal to 1 such that

$$U = D\tilde{U} \tag{6}$$

and the matrix A may be expressed uniquely as

$$A = LD\tilde{U} \tag{7}$$

If A is symmetric,

$$A = A^T = (LD\tilde{U})^T = \tilde{U}^T D L^T \tag{8}$$

This result implies that $\tilde{U}^T = L$ and A has an LDL decomposition:

$$\boxed{A = LDL^T} \tag{9}$$

3 Fun with "\"

a)

The linear system $Ax = b$, where $b \in \mathbf{R}^n$ and $A \in \mathbf{R}^{nn}$ where A is a nonsingular matrix may be solved simply with one line of Matlab code:

```
x = b ./ diag(A);
```
(10)

Since each element of x only requires one operation, the algorithm is linear in time and requires only $O(n)$ flops.

b)

Noise was added by perturbing A with βE with $\beta = 10^{-15}$ to obtain the following matrices:

1. $A_1 = A + \beta E_1$, where $E_1 = e_1 e_2^T$, and e_1, e_2 are the first two standard basis vector
2. $A_2 = A + \beta E_2$, where $E_2 = E_1 + E_1^T$
3. $A_3 = A + \beta E_3$, where E_3 generated by $E_3 = \text{triu}(\text{rand}(n))$;
4. $A_4 = A + \beta E_4$, where E_4 generated by $E_4 = \text{rand}(n)$; $E_4 = (E_4 + E_4)/2$;
5. $A_5 = A + \beta E_5$, where E_5 generated by $E_5 = \text{rand}(n)$

The test was done with 25 instances for $n = 1000$ for a random diagonal matrix A and vector b. The average and standard deviation of the time required to solve the unperturbed matrix using the algorithm of Equation 10 is compared with the backslash operator for each of perturbed matrices in Table 1.

The command mldivide took the least time to solve matrix A, slightly more time to solve matrices A_1 and A_3, much more time to solve matrices A_2 and A_4, and the most time to solve matrix A_5. These differences in time and computational complexity may be explained by the structure of the perturbed matrix and the default Matlab algorithm.

	average time [μs]	standard deviation [μs]
solve_diag(A)	50.36	26.9612
$A\backslash b$	940.84	306.3166
$A_1\backslash b$	983.2	181.2501
$A_2\backslash b$	19730.08	3057.426
$A_3\backslash b$	981.44	174.2838
$A_4\backslash b$	17857.24	3262.123
$A_5\backslash b$	29766.56	4557.8832

Table 1

The command mldivide must have detected the triangular structure of A, because the non-perturbed system took the least time to solve. The perturbed matrices A_1 and A_3 only took slightly longer because they are triangular and could be solved with back substitution, which is $O(n^2)$. Matrices A_2 and A_4 are symmetric, so the a Cholesky decomposition is used, which is $O(n^3)$. Matrix A_5 does not have any structure that may be exploited, so a full LU decomposition is used, which is approximately twice as complex as the Cholesky decomposition.

A Smetana_Gregory_1917370_A3_P3_DIARY.txt

```
run('Smetana_Gregory_1917370_A3_P3.m')
t_alg = 50.36(26.9612) [microsecond]
t_0 = 940.84(306.3166) [microsecond]
t_1 = 983.2(181.2501) [microsecond]
t_2 = 19730.08(3057.426) [microsecond]
t_3 = 981.44(174.2838) [microsecond]
t_4 = 17857.24(3262.123) [microsecond]
t_5 = 29766.56(4557.8832) [microsecond]
```

B Smetana_Gregory_1917370_A3_P3.m

```
%Smetana_Gregory_1917370_A2_P4
clear;
clc;
n = 1000;
t_alg = zeros(25,1);
t_0 = zeros(25,1);
t_1 = zeros(25,1);
t_2 = zeros(25,1);
t_3 = zeros(25,1);
t_4 = zeros(25,1);
t_5 = zeros(25,1);


for( i = 1:25);
    A = diag(rand(n,1));
    b = rand(n,1);

    tic;
    solve_diag(A,b);
    t_alg(i) = toc;

    tic;
    A\b;
    t_0(i) = toc;

    %% part 1
    beta = 10^-15;
    e1 = [1; zeros(n-1,1)];
    e2 = [0 ; 1; zeros(n-2,1)];
    E1 = e1 * e2';
    A1 = A + beta * E1;

    tic;
```

```
    A1\b;
    t_1(i) = toc;
    %% part 2
    E2 = E1 + E1';
    A2 = A + beta * E2;

    tic;
    A2\b;
    t_2(i) = toc;
    %% part 3
    E3 = triu(rand(n));
    A3 = A + beta * E3;

    tic;
    A3\b;
    t_3(i) = toc;
    %% part 4
    E4 = rand(n);
    E4 = (E4 + E4')/2;
    A4 = A + beta * E4;

    tic;
    A4\b;
    t_4(i) = toc;
    %% part 5
    E5 = rand(n);
    A5 = A + beta * E5;

    tic;
    A5\b;
    t_5(i) = toc;
end;

%% output results
disp(['t_alg = ',num2str(mean(t_alg*10^6)),...
    '(',num2str(std(t_alg*10^6)),') [microsecond]'])
disp(['t_0 = ',num2str(mean(t_0*10^6)),...
    '(',num2str(std(t_0*10^6)),') [microsecond]'])
disp(['t_1 = ',num2str(mean(t_1*10^6)),...
    '(',num2str(std(t_1*10^6)),') [microsecond]'])
disp(['t_2 = ',num2str(mean(t_2*10^6)),...
    '(',num2str(std(t_2*10^6)),') [microsecond]'])
disp(['t_3 = ',num2str(mean(t_3*10^6)),...
    '(',num2str(std(t_3*10^6)),') [microsecond]'])
disp(['t_4 = ',num2str(mean(t_4*10^6)),...
    '(',num2str(std(t_4*10^6)),') [microsecond]'])
disp(['t_5 = ',num2str(mean(t_5*10^6)),...
    '(',num2str(std(t_5*10^6)),') [microsecond]'])
```

C solve_diag.m

```
%Smetana_Gregory_1917370_A2_P4
function [ x ] = solve_diag( A, b )
%SOLVE_DIAG Solves the linear system Ax=b where A is nonsingular and
%diagonal

x = b ./ diag(A);

end
```