CMSC 660 HW IV

Joe Asercion

9/25/18

1 Chapter 5, Problem 1 (Modified)

The solution to $Au = b$ may be written $u = A^{-1}b$. Calculate the computational cost of finding B=inv(A).

1.1 a. Calculation

Show that about $(\frac{2}{3})n^3$ flops reduces $AB = I$ to $UB = L^{-1}$.

1.1.1 Answer

Starting with the assumption that we have the LU decomposition of matrix A, we can rewrite the initial problem statement as

$$\begin{aligned} AB &= I \\ LUB &= I \end{aligned}$$

It is known that $L = M_1^{-1}M_2^{-1}$[1](pg. 109). Inserting this definition into the above expression yields:

$$\begin{aligned} LUB &= I \\ M_1^{-1}M_2^{-1}UB &= I \\ \therefore UB &= M_2M_1 \end{aligned} \tag{1}$$

Using the same definition of L the expression for L^{-1} can derived:

$$\begin{aligned} L &= M_1^{-1}M_2^{-1} \\ L^{-1}L = I &= L^{-1}M_1^{-1}M_2^{-1} \\ M_2 &= L^{-1}M_1^{-1} \end{aligned}$$

$$M_2 M_1 = L^{-1} \tag{2}$$

Inserting equation (2) into equation (1) gives the final expression

$$UB = L^{-1}$$

As noted on [1](pg. 109), obtaining the inverse of the matrices M_1 and M_2 can be obtained using Gaussian Elimination. Therefore, since LU decomposition takes $\approx \frac{2}{3}n^3$ flops, the reduction of $AB = I$ to $UB = L^{-1}$ also takes $\approx \frac{2}{3}n^3$ flops.

1.2 b. Calculation

Show that computing the entries of B from $UB = L^{-1}$ by back substitution takes about n^3 flops.

1.2.1 Answer

Using the Back Substitution algorithm outline in the second program of part a,

$$\begin{aligned} \texttt{for i = n:1} \rightarrow & -\sum_{i=1}^{n} \\ \texttt{j = i+1:n} \rightarrow & \sum_{j=i+1}^{n} \\ \therefore W_n = & -\sum_{i=1}^{n} \Big(1 + \sum_{j=i+1}^{n} 2\Big) \end{aligned}$$

Converting to integrals and solving yields

$$\begin{aligned} W_n &= -\int_1^n \Big(1 + \int_{x+1}^{n} 2dy)dx \\ &= -\int_1^n (1 + 2n - 2x - 2)dx \\ &= n^2 - 3n + 2 \end{aligned}$$

The final operation which to obtain B, outlined in the code as

```
x(i) = rhs/U(i,i)
```

Contributes an additional n operations per element, therefore the total number of flops is approximately

$$F = n(n^2 - 3n + 2) \approx n^3$$

1.3 c. Proof

Use the above to verify the claim that computing A^{-1} is more than twice as expensive as solving $Au = b$ by LU factorization.

1.3.1 Answer

Part B shows that the cost of inverting a generic matrix scales as $\mathcal{O}(n^3)$, while Part A shows that LU decomposition scales as $\mathcal{O}(\frac{2}{3}n^3)$. Since taking the inverse of a matrix requires both forward and backward substitution, and forward substitution also takes n^3 flops, the total flops required to take the inverse of a matrix A is about $2n^3$, which is more than twice as expensive as the approximately $\frac{2}{3}n^3$ flops required for LU Factorization.

2 Chapter 5, Problem 4

A square matrix A has a bandwidth $2k + 1$ if $a_{jk} = 0$ whenever $|j - k| > k$. A subdiagonal or superdiagonal is a set of matrix elements on one side of the main diagonal with $j - k$, the distance to the diagonal, fixed. The bandwidth is the number of nonzero bands. A bandwidth 3 matrix is tridiagonal, bandwidth 5 makes pentadiagonal, and so on.

2.1 a. Calculation

Show that a SPD matrix with bandwidth $2k + 1$ has a Cholesky factor with nonzeros only on the diagonal and up to k bands below.

2.1.1 Answer

The Cholesky Factor, L, of a matrix A satisfies the expression $LL^* = A$. As an example, expanding the expression out in terms of the elements of the Cholesky Factor matrix, a $4x4$ SPD matrix A can be expressed as

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} & l_{11}l_{41} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 & l_{31}l_{21} + l_{32}l_{22} & l_{41}l_{21} + l_{42}l_{22} \\ l_{11}l_{31} & l_{21}l_{31} + l_{22}l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 & l_{41}l_{31} + l_{42}l_{32} + l_{43}l_{33} \\ l_{11}l_{41} & l_{21}l_{41} + l_{22}l_{42} & l_{31}l_{41} + l_{32}l_{42} + l_{33}l_{43} & l_{41}^2 + l_{42}^2 + l_{43}^2 + l_{44}^2 \end{bmatrix}$$

Solving for the individual elements of L yields the expressions:

$$l_{11} = \sqrt{a_{11}}$$
$$l_{21} = \frac{a_{21}}{l_{11}},\ l_{22} = \sqrt{a_{22} - l_{21}^2}$$
$$l_{31} = \frac{a_{31}}{l_{11}},\ l_{32} = \frac{a_{32}}{l_{22}} - \frac{l_{21}l_{31}}{l_{22}},\ l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2}$$
$$l_{41} = \frac{a_{41}}{l_{11}},\ l_{42} = \frac{a_{42}}{l_{22}} - \frac{l_{21}l_{41}}{l_{22}},\ l_{43} = \frac{a_{43}}{l_{33}} - \frac{l_{31}l_{41}}{l_{33}} - \frac{l_{32}l_{42}}{l_{33}},\ l_{44} = \sqrt{a_{44} - l_{41}^2 - l_{42}^2 - l_{43}^2}$$

In the case where $k = 1$, A has bandwidth 3. Therefore, A reduces to

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} = 0 & l_{11}l_{41} = 0 \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 & l_{31}l_{21} + l_{32}l_{22} & l_{41}l_{21} + l_{42}l_{22} = 0 \\ l_{11}l_{31} = 0 & l_{21}l_{31} + l_{22}l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 & l_{41}l_{31} + l_{42}l_{32} + l_{43}l_{33} \\ l_{11}l_{41} = 0 & l_{21}l_{41} + l_{22}l_{42} = 0 & l_{31}l_{41} + l_{32}l_{42} + l_{33}l_{43} & l_{41}^2 + l_{42}^2 + l_{43}^2 + l_{44}^2 \end{bmatrix}$$

Examining the zero terms of the matrix, it can immediately be noted from the first column and first row that the terms l_{31} and l_{41} must be 0, since $l_{11} \neq 0$. Further, it is clear that l_{42} must also be zero, since $l_{22} \neq 0$. Plugging this into the equations for the elements of L yields:

$$l_{11} = \sqrt{a_{11}}$$
$$l_{21} = \frac{a_{21}}{l_{11}},\ l_{22} = \sqrt{a_{22} - l_{21}^2}$$
$$l_{31} = 0,\ l_{32} = \frac{a_{32}}{l_{22}},\ l_{33} = \sqrt{a_{33} - l_{32}^2}$$
$$l_{41} = 0,\ l_{42} = 0,\ l_{43} = \frac{a_{43}}{l_{33}},\ l_{44} = \sqrt{a_{44} - l_{43}^2}$$

Therefore, in matrix form,

$$L = \begin{bmatrix} \sqrt{a_{11}} & 0 & 0 & 0 \\ \frac{a_{21}}{\sqrt{a_{11}}} & \sqrt{a_{22} - \frac{a_{21}^2}{a_{11}}} & 0 & 0 \\ 0 & \frac{a_{32}}{\sqrt{a_{22} - \frac{a_{21}^2}{a_{11}}}} & \sqrt{a_{33} - \frac{a_{32}^2}{a_{22} - \frac{a_{21}^2}{a_{11}}}} & 0 \\ 0 & 0 & \frac{a_{43}}{\sqrt{a_{33} - \frac{a_{32}^2}{a_{22} - \frac{a_{21}^2}{a_{11}}}}} & \sqrt{a_{44} - \frac{a_{43}^2}{a_{33} - \frac{a_{32}^2}{a_{22} - \frac{a_{21}^2}{a_{11}}}}} \end{bmatrix} \tag{3}$$

As seen in (3), the only elements of L which are non-zero are the diagonal and $k = 1$ rows below the diagonal. This pattern is true for any value of k, as the reduction and elimination of the l elements corresponding to the zero elements of A will inevitably zero out any L elements that are not within the diagonal or k rows below the diagonal.

2.2 b. Calculation

Show that the Cholesky decomposition algorithm computes this L in work proportional to $k^2 n$ if we skip operation on entries of A outside its nonzero bands.

2.2.1 Answer

Formally, the process used to calculate the matrix L above is the Cholesky-Banachiewicz Algorithm for Cholesky Decomposition[2]. In summary, the diagonal elements of the matrix L can be calculated using the formula

$$l_{j,j} = \sqrt{a_{j,j} - \sum_{k=1}^{j-1} l_{j,k}^2} \tag{4}$$

And the off-diagonal elements:

$$l_{i,j} = \frac{1}{l_{j,j}}(a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} l_{j,k}) \tag{5}$$

Combining (4) and (5) provides an expression for any arbitrary element of the matrix:

$$l_{i,j} = \frac{1}{\sqrt{a_{j,j} - \sum_{k=1}^{j-1} l_{j,k}^2}}(a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} l_{j,k}) \tag{6}$$

Indeed, when $i = j$ (6) reduces to (4).

The number of non-zero diagonal entries of A is n. The number of non-zero off-diagonal entries is dependent on k and can be expressed as $n_k = \sum_{i=1}^{k}(n-i)$.

Starting with (4), the total number of operations required to calculate the diagonal element of a particular row can be expressed as

$$N = (2 + \int_1^{n-1} dx)$$

$$N = (2 + (n-1) - 1) \approx n$$

The operations required to calculate the n_k off-diagonal elements can be expressed as

$$
\begin{aligned}
N_k &= n_k(2 + \int_1^{n-1} dx) \\
&= \int_1^k (2 + \int_1^{n-k} dy)dx \\
&= \int_1^k (1 + (n-k))dx \\
&= (k-1) + n(k-1) - k(k-1) = -k^2 + nk - n - 1
\end{aligned}
$$

Therefore, the total number of operations required to perform this Cholesky Decomposition is $n(k^2 - nk + n + 1) \approx k^2 n$ flops.

2.3 c. MATLAB

Write a procedure for Cholesky factorization of tridiagonal SPD matricies and apply it to the matrix of Exercise 4.11. Compare the running time with this dense matrix factorizer and the one from Exercise 5.4. Check that the answer is the same, up to roundoff.

2.3.1 Answer

MATLAB chol function:

```
%****************************
% CMSC660 HW4 Problem 2c
% Joe Asercion
%****************************

% Set iteration step size
delta_x=.1;

% Total number of steps
n=(1/delta_x);

% 'A' Transformation Matrix
A = (1/2)*(1/(delta_x)^2)*gallery('tridiag',n-1,1,-2,1);

% Note: Leaving A as a sparse matrix speeds up chol('lower') execution
L = chol(-A,'lower');
```

Execution Time: 0.022s (Time obtained from MATLAB Profiler)

Section 5.4 Algorithm:

```
%****************************
% CMSC660 HW4 Problem 2c
% Joe Asercion
%****************************

% Set iteration step size
delta_x=.1;

% Total number of steps
n=(1/delta_x);

diagSum = 0;
offdiagSum = 0;

% 'A' Transformation Matrix
A = -(1/2)*(1/(delta_x)^2)*gallery('tridiag',n-1,1,-2,1);

A = full(A);

L = zeros(size(A));

% Calculate L(1,1)
L(1,1) = sqrt(A(1,1));

% Use L(1,1) to obtain L(x,1)
for i = 2:n-1
  L(i,1) = A(i,1)/L(1,1);
end

%Calculate L(2,2)
L(2,2) = sqrt(A(2,2)-L(2,1)*L(2,1));

for i = 3:n-1
  for j = i-1:i
        if j == i
          L(j,j) = sqrt(A(j,j)-L(j,j-1)*L(j,j-1));
        else
          L(i,j) = A(i,j)/L(i-1,j);
    end
  end
```

Execution Time: 0.026s (Time obtained from MATLAB Profiler)

Both implementations returned the following values for L:

(1,1)	10.0000
(2,1)	−5.0000
(2,2)	8.6603
(3,2)	−5.7735
(3,3)	8.1650
(4,3)	−6.1237
(4,4)	7.9057
(5,4)	−6.3246
(5,5)	7.7460
(6,5)	−6.4550
(6,6)	7.6376
(7,6)	−6.5465
(7,7)	7.5593
(8,7)	−6.6144
(8,8)	7.5000
(9,8)	−6.6667
(9,9)	7.4536

3 Homework 4, Problem 3

Let A be a $N \times N$ symmetric matrix. Use Householder matrices to show that there exists an orthogonal matrix Q such that $T := Q^T AQ$ is tridiagonal.

3.0.1 Answer

We can use Householder matrices to transform the matrix A into the tridiagonal matrix T. The driving idea behind this process is the fact that Householder matrices (as expressions of Householder Transformations) reflect vectors in a matrix space across a unit hyperplane orthogonal to the vector's hyperplane. By applying Householder transformations to both the row and column vectors of matrix A systematically the matrix can be tridiagonalized.

Starting with an arbitrary $n \times n$ matrix A,

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & ... & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & ... & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & ... & ... \\ ... & ... & ... & ... & ... \\ a_{n,1} & a_{n,2} & ... & ... & a_{n,n} \end{bmatrix}$$

The first Householder Matrix needs to be selected such that

$$P_1 \to P_1 A = \begin{bmatrix} a'_{1,1} & a'_{1,2} & a'_{1,3} & ... & a'_{1,n} \\ a'_{2,1} & a'_{2,2} & a'_{2,3} & ... & a'_{2,n} \\ 0 & a'_{3,2} & a'_{3,3} & ... & ... \\ ... & ... & ... & ... & ... \\ 0 & a'_{n,2} & ... & ... & a'_{n,n} \end{bmatrix}$$

This matrix must also be applied to the row vectors. Therefore,

$$A_1 = P_1 A P_1 = \begin{bmatrix} a_{1,(1,1)} & a_{1,(1,2)} & 0 & ... & 0 \\ a_{1,(2,1)} & a_{1,(2,2)} & a_{1,(2,3)} & ... & a_{1,(2,n)} \\ 0 & a_{1,(3,2)} & a_{1,(3,3)} & ... & ... \\ ... & ... & ... & ... & ... \\ 0 & a_{1,(n,2)} & ... & ... & a_{1,(n,n)} \end{bmatrix}$$

A_1 can now be used to calculate the next Householder matrix P_2. Applying the same process in selecting P_2,

$$A_2 = P_2 A_1 P_2 = \begin{bmatrix} a_{2,(1,1)} & a_{2,(1,2)} & 0 & ... & 0 \\ a_{2,(2,1)} & a_{2,(2,2)} & a_{2,(2,3)} & ... & 0 \\ 0 & a_{2,(3,2)} & a_{2,(3,3)} & ... & ... \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & ... & a_{2,(n,n)} \end{bmatrix}$$

This process continues until the matrix $A_n = T$. This will require n Householder Matrices. This process can be expressed as:

$$\begin{aligned} T = A_n &= P_n P_{n-1} P_{n-2} ... P_3 P_2 P_1 A P_1 P_2 P_3 ... P_{n-2} P_{n-1} P_n \\ Q &:= P_1 P_2 P_3 ... P_{n-2} P_{n-1} P_n \\ \therefore Q^T &:= P_n P_{n-1} P_{n-2} ... P_3 P_2 P_1 \end{aligned}$$

Therefore,

$$T = Q^T A Q \tag{7}$$

References

[1] David Bindel and Johnathan Goodman. *Principles of Scientific Computing.* 2009.

[2] *Wikipedia Page for Cholesky Decomposition*

`https://en.wikipedia.org/wiki/Cholesky_decomposition`