

CS 475/CM 375 - Fall 2010: Assignment 1

Instructor: Pascal Poupart

Office: DC2514

Email: ppoupart@uwaterloo.ca

Classroom: MC 4060

TuTh 10:00am-11:20am

Office hours: TuTh 2:30-4pm

Class homepage: www.student.cs.uwaterloo.ca/~cs475/

Due: October 12, Tuesday (in class)

1. (10 marks) Given a dense symmetric matrix $a_{ij} = a_{ji}$, we need only store the upper triangular part of A , i.e.

$$\text{Upper}(A) = a_{ij} \quad j \geq i.$$

- (a) It is desired to factor this matrix so that

$$A = LU,$$

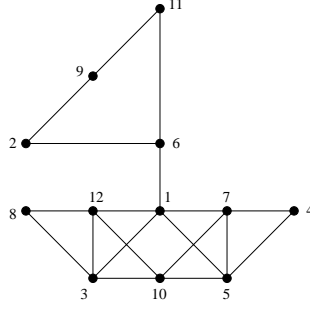
where L is unit lower triangular and U is upper triangular. Give precise pseudo code for converting the upper triangular part of A to U . At the end of the algorithm, the entries of A are replaced by the entries of U . This algorithm should operate *only* on the upper triangular part of A . (Hint: modify the pseudo code given in class for Gaussian elimination. Since A is symmetric, computation of L is not really necessary.)

- (b) Show that the complexity of the symmetric Gaussian elimination algorithm derived in part (a) is: $\frac{1}{3}n^3 + O(n^2)$, where n is the matrix size.
2. (10 marks) Suppose an $n \times n$ matrix A has an upper bandwidth q and lower bandwidth p . Suppose the L , U factors of A , i.e. $A = LU$, have already been computed and stored in the L and U matrices. Derive an *efficient* algorithm which will solve $Ax = b$. Give a precise pseudo code for your algorithm. What is the complexity of your algorithm? Show your work.
3. (10 marks) Given an $n \times n$ matrix A and its L , U factors, how do you *efficiently* solve the following problems using the L , U factors of A (no LU factorization of any matrix)? Note: you are not allowed to compute the inverse of A or any matrix.
- (a) Compute $\alpha = c^T A^{-1} b$.
- (b) Solve the matrix equation $Bx = b$ where $B = A + uv^T$, u, v are given $n \times 1$ vectors. (Hint: apply the Sherman-Morrison-Woodbury formula)
4. (5 marks) In a modified heat equation, the finite difference approximation results in a set of linear equations:

$$\alpha T_{i-1,j-1} + \beta T_{i,j-1} + \gamma T_{i+1,j-1} + \rho T_{i-1,j} + \delta T_{i,j} + \theta T_{i+1,j} + \mu T_{i-1,j+1} + \eta T_{i,j+1} + \nu T_{i+1,j+1} = f_{i,j},$$

$i, j = 1, 2, \dots, m$. Let A be the matrix of the linear system. Draw a picture of A . Describe the sparsity structure of A and the nonzero entries.

5. (10 marks) Consider a symmetric matrix whose graph is given by:



Perform Cuthill-McKee, reverse Cuthill-McKee, and minimum degree orderings on this matrix. For minimum degree ordering, ties are broken by selecting the node with smaller original node number (as shown on the graph).

6. (40 marks) In modeling of heat flow, the temperature $T = T(x, y)$ satisfies the Poisson equation

$$-\frac{\partial^2 T}{\partial x^2} - \frac{\partial^2 T}{\partial y^2} = f(x, y),$$

where $f(x, y)$ is the heat source function.

We approximate $T(x, y)$ at discrete locations on a two dimensional grid. Let the (i, j) grid point have location (x_i, y_j) where $x_i = ih$, $y_j = jh$, and $h = 1/(m + 1)$ is the grid size. Let $T_{i,j} \approx T(x_i, y_j)$. Then the finite difference approximation results in a set of linear equations

$$\frac{1}{h^2}(4T_{i,j} - T_{i-1,j} - T_{i+1,j} - T_{i,j-1} - T_{i,j+1}) = f_{i,j}. \quad (1)$$

(Note: we assume the boundary temperatures at the sides of the grid are zero.) We want to analyze the heat flow with two central heating systems; i.e.

$$f_{i,j} = \begin{cases} 1 & \text{if } \|(x_i, y_j) - (0.3, 0.3)\| \leq 0.1 \\ 1 & \text{if } \|(x_i, y_j) - (0.6, 0.6)\| \leq 0.1 \\ 0 & \text{otherwise.} \end{cases}$$

An example of temperature distribution with four heat sources is shown in Figure 1.

Define a vector x such that

$$x_k = T_{i,j},$$

where $k = (j - 1)m + i$. Similarly, we define the vector b with $b_k = f_{i,j}$. Then we can write equation (1) in the matrix form

$$Ax = b. \quad (2)$$

The coefficient matrix A is sparse of size $n \times n$, where $n = m^2$.

- (a) Create a MATLAB function:

$$[A, b] = \text{Lap2D}(m)$$

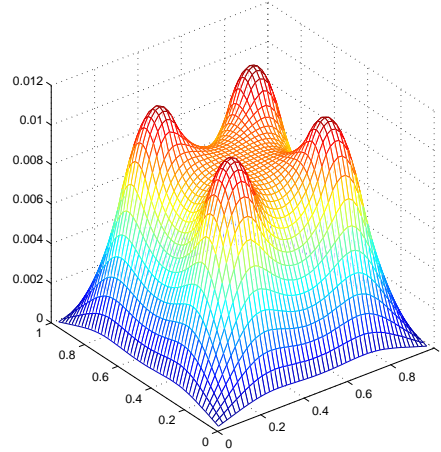


Figure 1: Temperature distribution with four heat sources.

The input is m , the number of grid points in each dimension. The outputs are the matrix A and the right-hand side b of (2).

- (b) Implement the numerical methods: Gaussian elimination, Cholesky factorization, and Band Gaussian elimination. Create the following MATLAB functions:

```
x = GaussElim(A,b)
x = Cholesky(A,b)
x = BandGE(A,b)
```

These MATLAB functions take as inputs the matrix A and right-hand side b and compute the solution x using the corresponding variant of Gaussian elimination method. No need to implement pivoting. Also, do not check for zero or small pivots.

You may check your code by viewing the solution x . Download the MATLAB function `x1to2.m` from the class homepage. Do the following in MATLAB

```
>> mesh(x1to2(x,m,m));
```

It will generate a 2D mesh plot of the solution x . Submit a 2D mesh plot for $m = 20$.

- (c) Create a MATLAB script, `GETimes`, that solves (2) using different variants of Gaussian elimination. In particular, set up the matrix A and right-hand side b by calling the MATLAB function `Lap2D`. Then solve the equation by calling one of the MATLAB functions in part (b). Record the CPU time using the MATLAB command `cputime`. Construct a table of execution times for solving (2) using Gaussian elimination, Cholesky factorization and Band Gaussian elimination. Try the grid sizes: 10×10 , 20×20 , 30×30 , and 40×40 . Compare and comment on the timing results.

Submit: `Lap2D.m`, `GaussElim.m`, `Cholesky.m`, `BandGE.m`, `GETimes.m`, a mesh plot of x , a table of cpu times, comments on the timing results.

7. (10 marks) Ordering methods and number of fill-in's.

- (a) Modified `Lap2D` and create another MATLAB function:

```
A = ReorderLap2D(m,ordering_method)
```

The first input parameter specifies the grid size. The second input, `ordering_method`, is a string that can be one of the following: `'natural'`, `'rcm'`, `'min'`, which correspond to the natural, RCM, and minimum degree orderings, respectively. In order to construct the ordering vector for RCM, do the following:

```
>> p = symrcm(A);  
>> B = A(p,p);
```

The matrix B is the reordering of A using the RCM ordering. The MATLAB command for the minimum degree ordering is `symamd`. Use `sparse` to convert a dense matrix A to a sparse matrix. The input for both `symrcm` and `symamd` must be a sparse matrix.

- (b) Create a MATLAB script, `CompareNNZ`, that compares the number of nonzeros in the L , U factors of A using different orderings. The program calls `ReorderLap2D` to create a sparse matrix A . Then compute the L , U factors using the MATLAB command `lu`. Determine the number of nonzeros of L using the MATLAB command `nnz`. (Note: the number of nonzeros of L and U are the same since A is symmetric). Construct a table of number of nonzeros in A and L for different ordering methods and different grid sizes. The grid sizes are: 20×20 , 40×40 , 80×80 , 160×160 , 320×320 . Comment on the number of nonzeros in different cases. In particular, how are the number of nonzeros affected by the ordering methods and the grid dimensions? Which ordering method produces the least number of nonzeros in L ?

Submit: `ReorderLap2D.m`, `CompareNNZ.m`, a table of number of nonzeros, comments on the results.