CS472/CM472/CS672 - Fall 2007: Assignment 1

Instructor: Pascal Poupart | Office: DC2514 | Email: `ppoupart@cs.uwaterloo.ca`
Classroom: MC4042 | Tu,Th 4:00-5:20 | Office Hours: Wed; 10:00-11:00
Web Site: www.student.cs.uwaterloo.ca/ ~cs472/ | Newsgroup: uw.cs.cs472

TA: Zhuliang Chen | Email: `z4chen@cs.uwaterloo.ca`
Special office hours: Monday, Oct 1, 2-3pm (room TBA)

Due: Tuesday, October 2nd (at the beginning of class)

The main objective of this assignment is to familiarize yourself with the ia-ja data structures. The matrix of interest is the 2D Laplacian arising from heat flow problems. You should find this assignment fairly straightforward.

A model of heat flow is given by

$$q_x = -\frac{\partial T}{\partial x}, \quad q_y = -\frac{\partial T}{\partial y},$$

where (q_x, q_y) is the heat flow velocity, and $T = T(x, y)$ is the temperature which satisfies the Poisson equation

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

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

We approximate the temperature function $T(x, y)$ 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 $h = 1/(n + 1)$ 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

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

Solve (1) on an $n \times n$ grid with $i = 1, \ldots, n$; $j = 1, \ldots, n$. (Note: we assume the boundary temperature at the sides of the grid are zero.) We want to analyze the heat flow with multiple central heating systems as shown in Figure 1; i.e.

$$f_{i,j} = \begin{cases} 1 & \text{if} \quad \|(x_i, y_j) - (0.25, 0.25)\|_2 \le 0.1 \\ 1 & \text{if} \quad \|(x_i, y_j) - (0.25, 0.75)\|_2 \le 0.1 \\ 1 & \text{if} \quad \|(x_i, y_j) - (0.75, 0.25)\|_2 \le 0.1 \\ 1 & \text{if} \quad \|(x_i, y_j) - (0.75, 0.75)\|_2 \le 0.1 \\ 0 & \text{otherwise.} \end{cases}$$

Define a vector x such that

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

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

$$Ax = b.$$

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

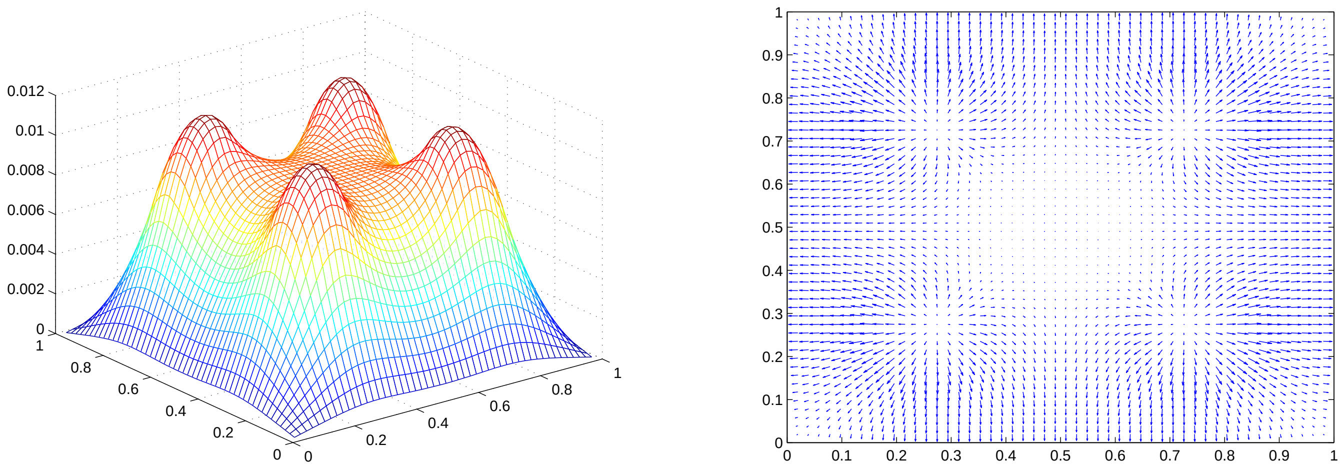

Figure 1: (a) Temperature function (b) Heat flow with four heat sources.

1. (10 marks) Set up the matrix A and the right-hand side (RHS) b. Use ia-ja data structure for A and an 1D array (i.e. full vector) for b. The main program, *main.c*, can be downloaded from the class homepage. It allocates appropriate memory for the arrays `ia`, `ja` and `a` as well as the RHS array `b`. Then it calls the functions `matrix` and `source` to set up the values for A and b, respectively. Your job is to complete the body of these functions in the file *matrhs.c*. (Note: in C, indices start from 0.)

   In order to black box test your code, place your version of *matrhs.c, matrhs.h* in a single directory (nothing else in this directory) with the name

   ```
   your_userid_your_student_id
   ```

   Then zip up this directory using

   ```
   zip -r your_userid_your_student_id your_userid_your_student_id
   ```

   Mail the file

   ```
   your_userid_your_student_id.zip
   ```

   (as an attachment) to `z4chen@cs.uwaterloo.ca`. Your code will be tested by linking your code with the *main.c* calling routine (*main.o*), and running some tests. Submit a hard copy listing of the code as well.

2. (4 marks) To test your code, for the case $n = 50$, uncomment the part of the code marked "Dump data" in *main.c*. (Note: it automatically converts the index range from 0 to $N-1$ in C to 1 to N in MATLAB.) It will then save all data to the file *output.m*. The MATLAB program `heatflow.m` reads the data from *output.m*, solves the matrix equations and finally displays the temperature and heat flow of your computed solution. Submit the plots.

3. (6 marks) In a modified heat equation, the finite difference approximation results in a set of linear equations:

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

Let A be the matrix of the linear system. Describe the sparsity structure of A and the nonzero entries.

4. (6 marks) Consider solving the linear system: $Ax = b$ where

$$A = \begin{bmatrix} 1 & 1 & 0 & 3 \\ 2 & 1 & -1 & 1 \\ 3 & -1 & -1 & 2 \\ -1 & 2 & 3 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 4 \\ 1 \\ -3 \\ 4 \end{bmatrix}.$$

Let L, U be the LU factorization of A, and [L\U] be the matrix whose upper triangular part is U and lower triangular part is the same as the lower traingular part of L.

   (a) Compute the [L\U] matrix using the standard Gaussian elimination. Show all your steps.
   (b) Compute the [L\U] matrix using the Doolittle algorithm. Show all your steps.
   (c) Using the L, U factors computed from (a) (or (b)), compute the solution x.

5. (8 marks) Given a dense symmetric matrix $a_{ij} = a_{ji}$, we need only store the upper triangular part of A, i.e.

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

It is desired to factor this matrix so that

$$A \quad = \quad LU,$$

where L is unit lower triangular. Give precise pseudo code for converting an upper triangular row i of A into row i of U (Doolittle form). For simplicity you may assume A and U are stored in separate matrices. This algorithm should operate only on the upper triangular part of A. (Hint: modify the pseudo code given in class for the Doolittle algorithm.) Then, give precise pseudo code (e.g., pseudo code for forward solve and backward solve) for solving

$$LU\,x \quad = \quad b,$$

by using only elements of U. (Hint: for symmetric A, we can factor $A = LDL^T$, where L=lower triangular, unit diagonal and D=diagonal. Thus $U = DL^T$, or equivalently, $L = U^T D^{-1}$.)

6. (10 marks) **Graduate student question**. Suppose we have an ia-ja representation of the nonzero structure of a matrix having symmetric structure. However, the list for each row is unsorted. Assuming the number of nonzeros, $nnz(A) = O(N)$, describe an algorithm for sorting each of these lists in worst case complexity $O(N)$. Note that there is no $O(NlogN)$ term in this complexity, even if there is a dense row.

Hint: Let C_i be the set of unsorted column indices for row i. Let C_i^{sorted} be the sorted column indices for row i. Given C_i, $i = 1, \ldots, N$, then we can construct C_i^{sorted}, $i = 1, \ldots, N$ in the following way. Allocate C_i^{sorted} of the correct size, $i = 1, \ldots, N$. Then scan through the rows in the order $i = 1, \ldots, N$. At row i, if $j \in C_i$, add column index i to C_j^{sorted}.

Implement the sorting algorithm. Submit a listing of your code, and the results of a small test case from question 1. Name the sorting subroutine as `symsort` with the following prototype:

```
symsort(int *ia, int *ja, double *a, int N).
```

Include it in *matrhs.c*.

Practice Problems (Do not hand in)

1. Show that if A is diagonally dominant by rows, then so is $A^{(k)}$.

2. Describe an algorithm for computing the LU factorization of a matrix so that U is unit upper triangular.

3. The Minimum Degree strategy for determining an ordering of nodes for Gaussian Elimination is the following

   - At each stage of the elimination, select the node which has the smallest degree (the degree of a node is the number of its graph neighbours). Ties are broken arbitrarily.

   Given a matrix with symmetric structure, whose graph is a tree, use the graph model of Gaussian elimination to show that minimum degree ordering generates a perfect elimination sequence (no fill-in is produced).