Lab Course
Scientific Computing
Worksheet 3

Alfredo Parra
Shulin Gao

Due December 5th, 2011

For this worksheet we study the two-dimensional stationary heat equation, namely

$$T_{xx} + T_{yy} = -2\pi^2 \sin \pi x \sin \pi y \tag{1}$$

on the unit square $]0;1[^2$ and Dirichlet boundary conditions

$$T(x,y) = 0 \forall (x,y) \in \partial]0;1[^2. \tag{2}$$

The exact solution is given by

$$T(x,y) = \sin \pi x \sin \pi y. \tag{3}$$

a) For the partial derivatives, we choose the following finite difference discretization:

$$T_{xx}|_{i,j} \approx \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{h_x^2}, \tag{4a}$$

$$T_{yy}|_{i,j} \approx \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{h_y^2}. \tag{4b}$$

Inserting these approximations into Eq. 1, we get

$$\frac{T_{i-1,j}}{h_x^2} + \frac{T_{i,j-1}}{h_y^2} - 2\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right) T_{i,j} + \frac{T_{i+1,j}}{h_x^2} + \frac{T_{i,j+1}}{h_y^2} = -2\pi^2 \sin(\pi x) \sin(\pi y), \tag{5}$$

which we can write in matrix form as follows:

$$\begin{pmatrix} -2\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right) & \frac{1}{h_x^2} & \cdots & \frac{1}{h_y^2} & & & & & \\ & \ddots & \ddots & & \ddots & & & & \\ \frac{1}{h_y^2} & & \cdots & \frac{1}{h_x^2} & -2\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right) & \frac{1}{h_x^2} & \cdots & \frac{1}{h_y^2} & \\ & \ddots & & \ddots & & \ddots & \ddots & & \ddots \\ & & & & \frac{1}{h_y^2} & & \cdots & \frac{1}{h_x^2} & -2\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right) \end{pmatrix} \begin{pmatrix} T_{1,1} \\ T_{2,1} \\ \vdots \\ T_{i,j} \\ \vdots \\ T_{N_x-1,N_y} \\ T_{N_x,N_y} \end{pmatrix} = -2\pi^2 \begin{pmatrix} \sin(\pi x_1)\sin(\pi y_1) \\ \sin(\pi x_2)\sin(\pi y_1) \\ \vdots \\ \sin(\pi x_i)\sin(\pi y_j) \\ \vdots \\ \sin(\pi x_{Nx})\sin(\pi y_{Ny}) \end{pmatrix}, \tag{6}$$

where the horizontal dots on the matrix indicate a distance of N_x to the right or to the left of the main diagonal. In other words, we have $-2\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right)$ on the main diagonal; then, on the upper and lower adjacent diagonals we have $\frac{1}{h_x^2}$, except for the rows corresponding to any $T_{1,j}$ (for which $T_{0,j} = 0$) and $T_{N_x,j}$ (for which $T_{N_x+1,j} = 0$); finally, if we move N_x places to the right and to the left of the main diagonal, we have $\frac{1}{h_y^2}$. The rest of the entries are zero.

b) The function that creates this matrix in MATLAB is implemented in the file `createM.m`. It takes N_x and N_y as parameters.

c) The Gauss-Seidel method is implemented in the MATLAB file `GaussSeidel.m`. The idea is the following. We have a system of $N_x \cdot N_y$ equations with $N_x \cdot N_y$ unknowns. We start with an initial guess for the vector $T_{i,j}$, which will be $T_{i,j} = 0$ for this exercise. With this guess, we calculate $T_{1,1}$. Using this newly calculated value, we proceed to calculate $T_{2,1}$, and so on. At every step k, we calculate the local residual $R_k = b_k - \sum_m a_{k,m} x_m$. The residual norm for the whole matrix system is then given by

$$R = \sqrt{\frac{1}{N} \sum_k R_k^2} = \sqrt{\frac{1}{N_x \cdot N_y} \sum_k \left(b_k - \sum_m a_{k,m} x_m \right)^2}. \tag{7}$$

d) The system 6 was solved using (1) the Gauss-Seidel scheme, (2) the full matrix system, and (3) the matrix system using a sparse matrix. The corresponding MATLAB files are `GaussSeidel.m` and `MatrixSolve.m` (commenting or uncommenting the `sparse()` command in the code, respectively).

e) This section contains the plots (3D and contour) for each of the three methods with $N_x = N_y = 7, 15, 31, 63$. The exact solution is seen in Fig. 1.

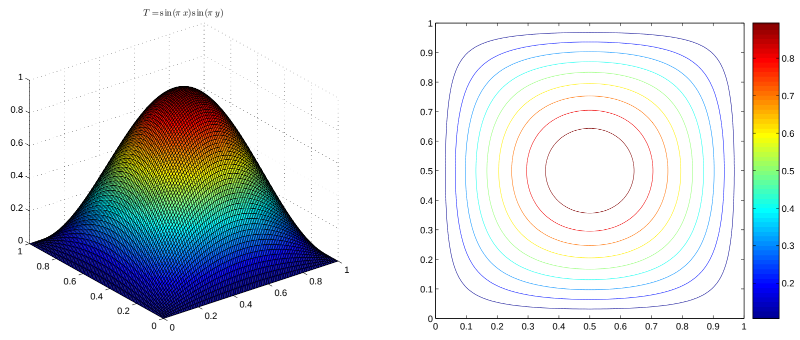

Figure 1: Exact solution of Eq. (1) using $N_x = N_y = 100$.

The next plots correspond to the solutions using the Gauss-Seidel method.

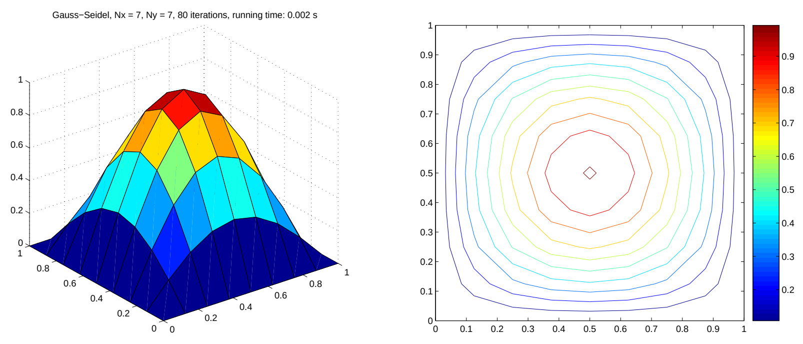

Figure 2: Gauss-Seidel plot for $N_x = N_y = 7$.

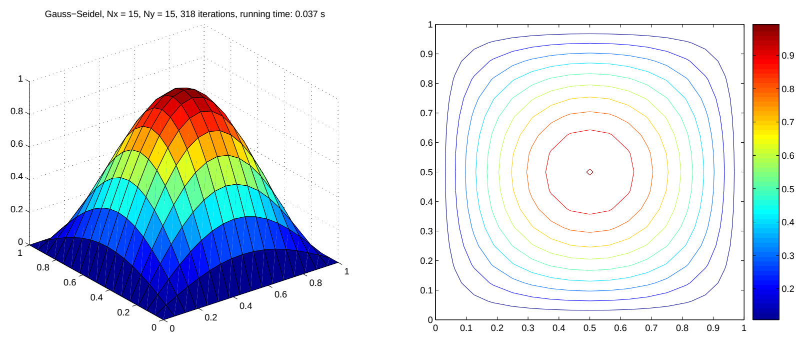

Figure 3: Gauss-Seidel plot for $N_x = N_y = 15$.

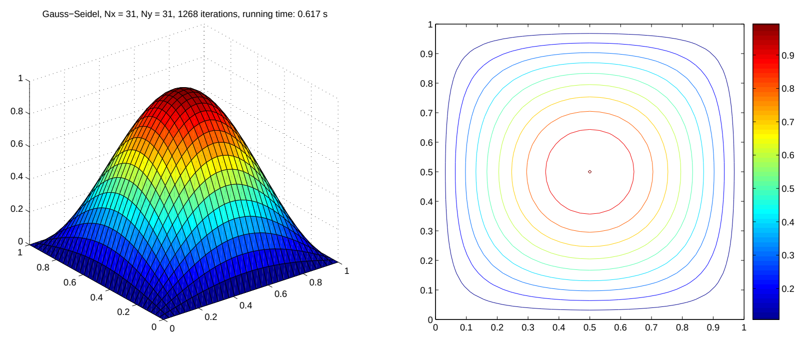

Figure 4: Gauss-Seidel plot for $N_x = N_y = 31$.

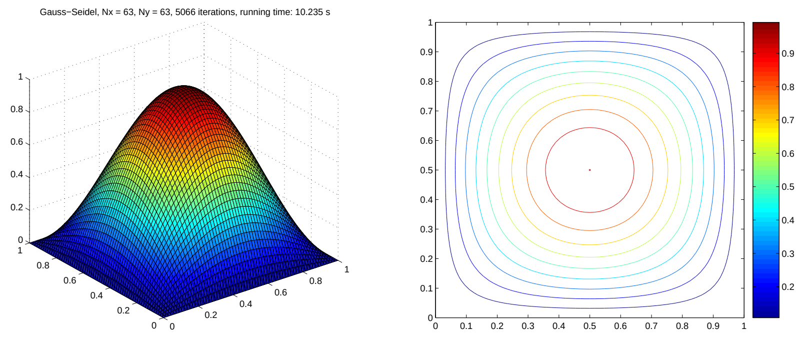

Figure 5: Gauss-Seidel plot for $N_x = N_y = 63$.

Now we observe the solutions using the full matrix system.

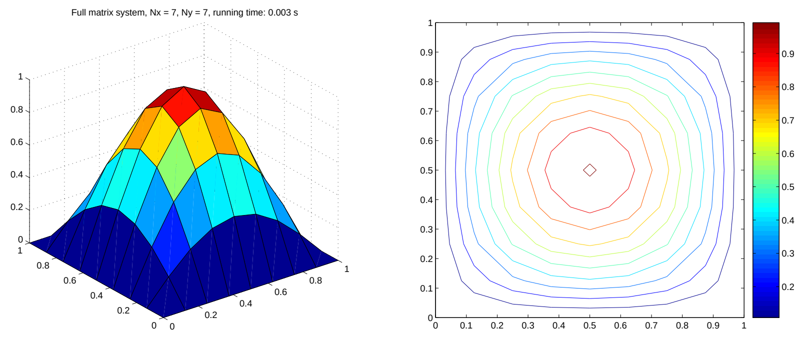

Figure 6: Full matrix system plot for $N_x = N_y = 7$.

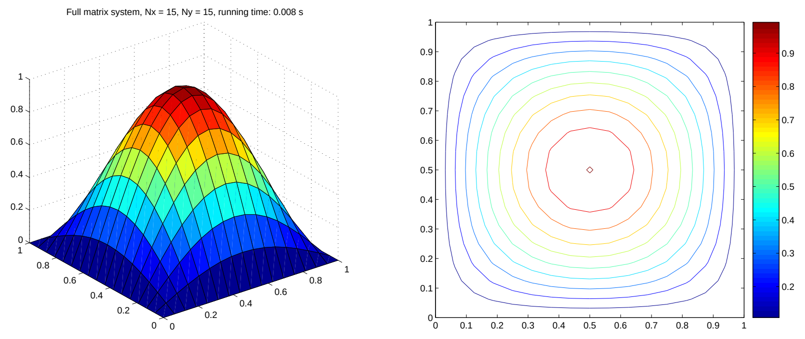

Figure 7: Full matrix system plot for $N_x = N_y = 15$.

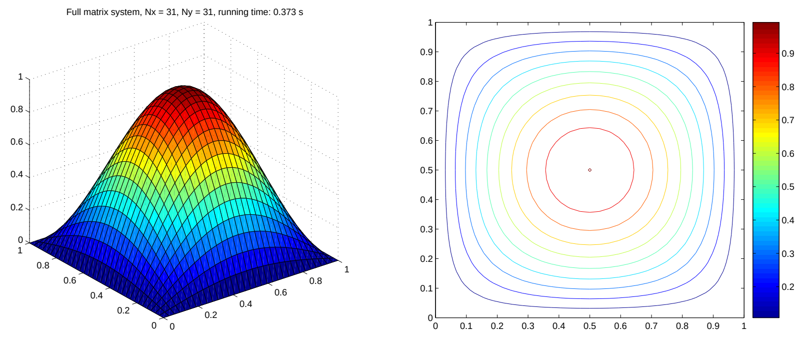

Figure 8: Full matrix system plot for $N_x = N_y = 31$.

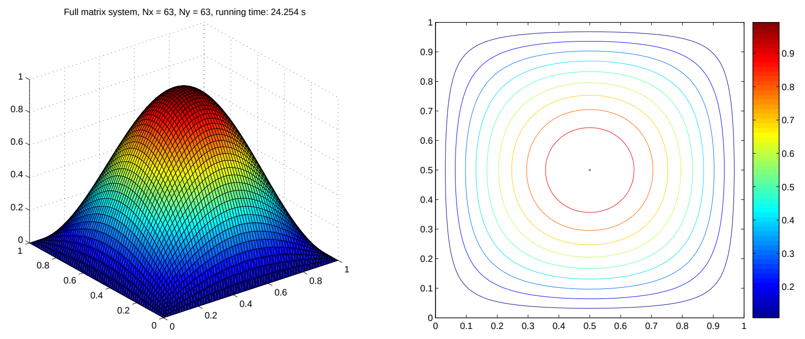

Figure 9: Full matrix system plot for $N_x = N_y = 63$.

Finally, we show the plots using the sparse matrix form of the system 6.

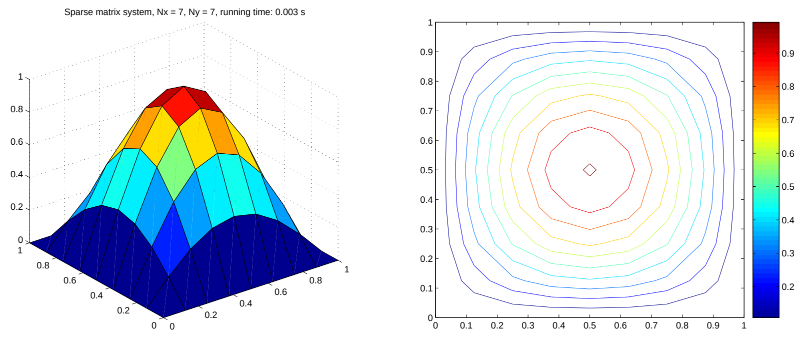

Figure 10: Sparse matrix plot with $N_x = N_y = 7$.

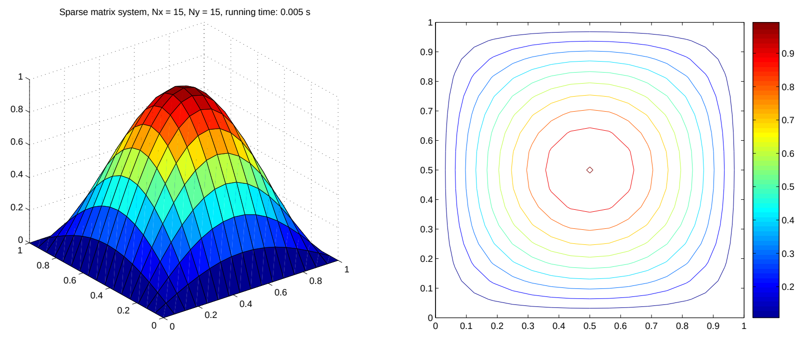

Figure 11: Sparse matrix plot with $N_x = N_y = 15$.

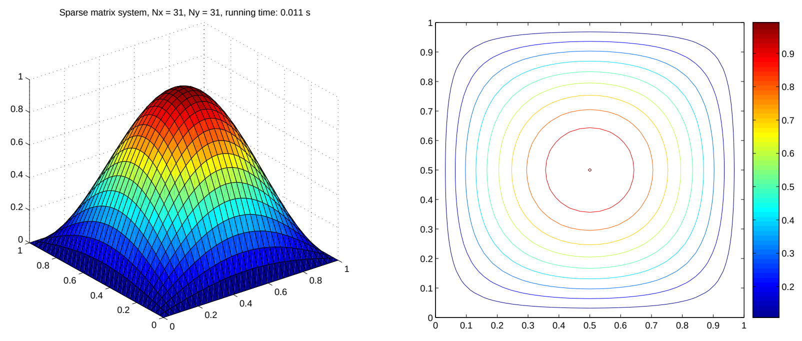

Figure 12: Sparse matrix plot with $N_x = N_y = 31$.

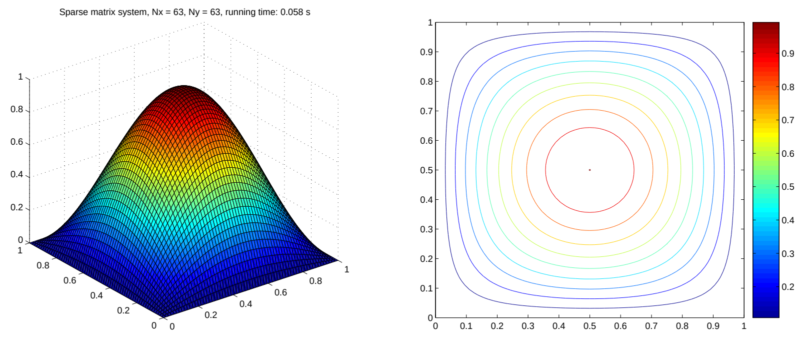

Figure 13: Sparse matrix plot with $N_x = N_y = 63$.

f) The runtimes and the storage requirements are different for all the three methods. Let's begin with the Gauss-Seidel method. The runtimes are shown in the table below. We are interested in knowing the number of entries stored in the memory to implement the method. Since each $T_{i,j}$ can be calculated using other values within the T vector and a function evaluation, we need

$length(T) = N_x \cdot N_y$ stored entries. The rest of the variables needed are constant: the x and y values to evaluate $\sin(\pi x)\sin(\pi y)$ at each (i, j), the values of N_x and N_y, the j index, the value of the residual, etc. This constant is of the order of 10^1 and can be ignored for our analysis.

Iterative solution with Gauss-Seidel				
N_x, N_y	7	15	31	63
Runtime (s)	0.002	0.037	0.617	10.235
Storage (# entries)	49	225	961	3969

The runtimes using the full matrix system are reported in the table below. For this case, a $(N_x \cdot N_y) \times (N_x \cdot N_y)$ matrix is stored in the memory, as well as the RHS vector with $N_x \cdot N_y$ entries and eventually the T solution vector, also with $N_x \cdot N_y$ entries. The values at the boundaries require another $2(N_x + N_y + 2)$ storage entries. In total, $N_x^2 N_y^2 + 2(N_y N_x + N_x + N_y + 2)$ entries have to be stored in memory, plus a constant which is of the order of 10^1 (which we ignore). So, if $N_x = N_y = N$, the entries are in $O(N^4)$.

Direct solution with full matrix				
N_x, N_y	7	15	31	63
Runtime (s)	0.003	0.008	0.373	24.254
Storage (# entries)	2531	51139	925571	15761155

Storing the matrix as a sparse matrix gives some advantages both in runtime and storage, since MATLAB only stores the non-zero entries with their corresponding indices. Since the matrix in (6) has $5N_xN_y - 2(N_x + N_y)$ nonzero entries (see question 1), MATLAB needs to store two indices per entry, plus the RHS vector of length $N_x \cdot N_y$ and the solution vector T of the same length. The total sum adds up to $17N_xN_y - 6(N_x + N_y)$, so the number of entries is in $O(N^2)$ plus a negligible constant. The runtime and number of storage entries can be observed in the following table:

Direct solution with sparse matrix				
N_x, N_y	7	15	31	63
Runtime (s)	0.003	0.005	0.011	0.058
Storage (# entries)	749	3645	15965	66717

g) A measure of the error for the Gauss-Seidel method can be estimated by the expression

$$e = \sqrt{\frac{1}{N_x \cdot N_y} \sum_{j=i}^{N_y} \sum_{i=1}^{N_x} (T_{i,j} - T(x_i, y_j))^2}. \tag{8}$$

In the next table we show the error and the reduced error for different choices of N_x and N_y

$N_x = N_y$	7	15	31	63	128
error	7.399e-3	1.714e-3	4.123e-4	9.948e-5	2.277e-5
error red.	-	0.2317	0.2405	0.2413	0.2289

Questions

1) The number of non-zero entries of the matrix in (6) is the sum of the terms of the five diagonals. The main diagonal is always non-zero, and there it has $N_x \cdot N_y$ terms. There exist N_y terms which vanish for $T_{i-1,j}$ (namely, $T_{1,1}$, $T_{1,2}$,..., T_{1,N_y}), and the same for $T_{i+1,j}$. These two diagonals have then $N_x \cdot N_y - N_y$ non-zero terms each. Likewise, there are N_x terms that vanish for $T_{i,j-1}$ and $T_{i,j+1}$, so each account for $N_x \cdot N_y - N_x$ non-zero entries. This gives a total of $\boxed{5N_x \cdot N_y - 2(N_x + N_y)}$ non-zero entries.

2) A full matrix of the same size would have $(N_x \cdot N_y)^2$ non-zero entries, so the proportion between such a matrix and our matrix is equal to $\chi = 5/N_x \cdot N_y - 2(N_x + N_y)/(N_x \cdot N_y)^2$, which goes to zero very fast when N_x and N_y become large. Since all the zero entries contain no information about the solution, the matrix method takes up too much unnecessary memory, thus making it inefficient in terms of storage.

3) Although the runtimes are comparatively smaller using the sparse matrix system, the memory requirements are very high. Indeed, we observed that for $N_x = N_y > 118$, our computers run out of memory to save the entries, but it is very fast up until that dimension. The Gauss-Seidel method does not have this issue, so for large N_x and N_y, this would be the best alternative. The full matrix system crashes when $N_x = N_y > 99$ (taking 351 seconds to solve the largest system).

4) For partial differential equations, the error in general is specified for each of the independent variables. For our problem, this means that the error is of order p in x and of order q in y if the error behaves like $O(h_x^p + h_y^q)$. Since our equation treats the x and y coordinates equally, we can talk about an error of order p behaving like $O(h^p)$. Since the reduced error seems to approximate the value of 1/4, we assume the error is of order 2, which is easy to prove using the Taylor expansion of the terms in the discretizations (4).