Fourth Homework

A0109983 Liu Zhaoqiang

April 17, 2015

Problem 1: The code is in finalProject_1.cpp. And the result for n=100000000 are shown in problem1.png.

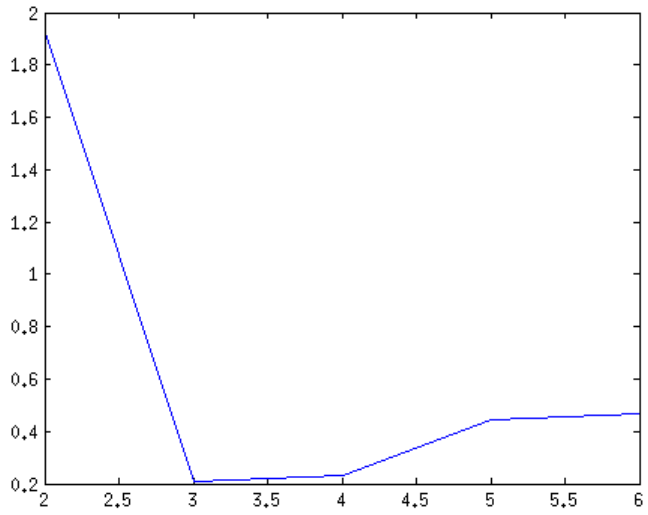
Problem 2: In this problem, we use both sequential and parallel jacobi iteration to solve:

$$u''(x) = f(x), u(0) = y_1, u(1) = y_2, x \in [0, 1]$$

and we set $u(x) = cos(x)e^x$, thus $f(x) = -2sin(x)e^x$.
The sequential code is in jacobi_serial.cpp, and the parallel code is in jacobi_parallel.cpp. To verify the correctness of the solvers, we set n=101(n is the number of small intervals, and the number of grid points we need to calculate is n-1=100), and print the errors of first 10 iterations for both sequential and parallel codes. The results are shown in problem2.png.

Problem 3: Because in the numerical test we found that if we wanted to get a not bad numerical solution, we must set a very small error_limit(we do not stop iteration until $||x^{(k+1)} - x^{(k)}||_\infty < error\_limit$) and iterate too many times. For example, when number of grid points is 100000, if we set error_limit=10^{-6}, after 1 million iterations, the numerical result is still very bad when comparing with the exact solution. Therefore, because we just need to comparing the running time, to save computation time, we let the maximum number of iteration be 10000.
For the serial code, to compare with the parallel code, we just use this compiler option: g++ jacobi_serial.cpp -o jacobi_serial, i.e, without any optimization option. To record the time of parallel computing, using start=MPI_Wtime() after a MPI_Barrier(comm), then using MPI_Wtime() again for a rank after the whole iteration. To reduce the computation time, because our maximum iteration times is given, we can use MPI_Reduce() instead of MPI_Allreduce(). Run each case for 50 times(e.g. mpirun -np 2 ./jacobi_parallel 10001), and select the minimum running time. But even though we have runned many times, due to the instablility of the enviroment. The running time still can be very strange. Table for the running time(ms) are as follows:

Figure 1: ratio of 2 nodes run times to the serial code run time



k	**np=1**	**np=2**	**np=4**
2	20	38.641	316.891
3	270	54.311	240.596
4	2700	631.853	3222.35
5	34040	15222.3	31302.6
6	350440	164610	168357

The graph of the ratio of my parallel run times to the serial code run time are as follows;

Because when k=2, the communication time seems to be largely more than the comutation time, and make the ratio largely different with k=3:6. We plot the ration for k=3:6 below.

Problem 4: To implement a parallel Gauss-Seidel method in two dimensions, we can use Red-Black Ordering. The point (i, j) in the mesh is colored according to the value of i + j: If i + j is even, then the mesh point is red, and if i + j is odd, then the mesh point is black. The goal of the reordering is to get an equivalent equation system in which more independent computations exist and, thus, a higher potential parallelism results. The points in the grid now form two sets of points. Both sets are numbered separately in a rowwise way from left to right. First the red points are numbered by $1, ..., n_R$, where n_R is the number of red points. Then, the black points are numbered by $n_{R+1}, ..., n_R + n_B$ where n_B is the number of black points and $n = n_R + n_B$. Then there are n_R unknowns associated with the red points

Figure 2: ratio of 4 nodes run times to the serial code run time

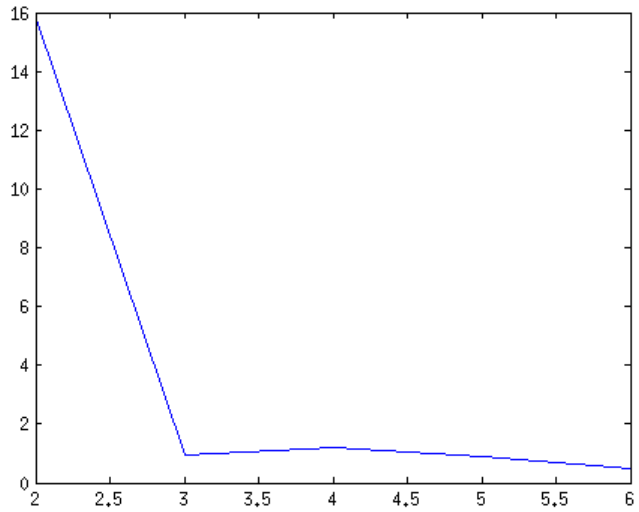


Figure 3: ratio of 2 nodes run times to the serial code run time(k=3:6)

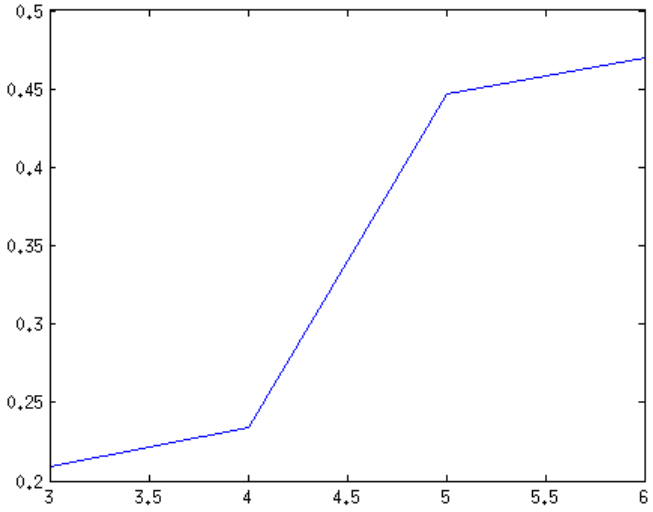
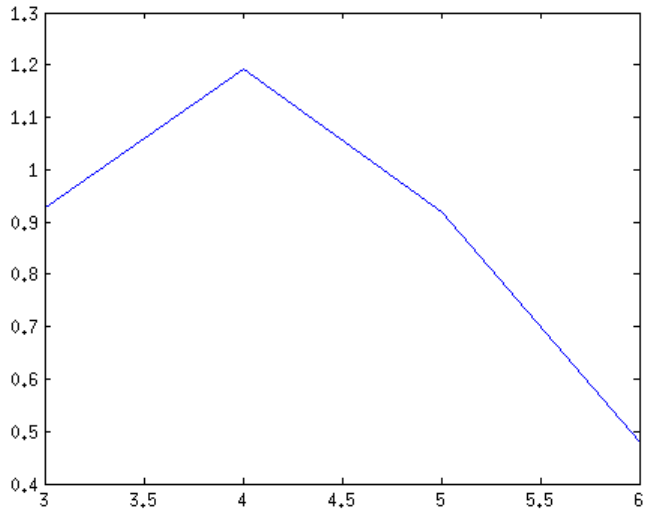


Figure 4: ratio of 4 nodes run times to the serial code run time(k=3:6)



denoted as $\hat{x}_1, ..., \hat{x}_{n_R}$, and n_B unknowns associated with the black points denoted as $\hat{x}_{n_R+1}, ..., \hat{x}_{n_R+n_B}$. Then the initial equation $Ax = b$ becomes $\hat{A}\hat{x} = \hat{b}$, where $\hat{A}$ is

$$\begin{pmatrix} D_R & F \\ E & D_B \end{pmatrix}$$

and $\hat{x} = (\hat{x}_R, \hat{x}_B)^T$, $\hat{b} = (\hat{b}_R, \hat{b}_B)^T$. $\hat{x}_R$ denotes the subvector of size n_R of the red unknowns and $\hat{x}_B$ denotes the subvector of size n_B of the black unknowns. The right-hand side b of the original equation system is reordered accordingly. The submatrices D_R and D_B are diagonal matrices and the submatrices E and F are sparse banded matrices. Using Gauss-Seidel method for the new linear equation, we have:

$$D_R \hat{x}_R^{(k+1)} = \hat{b}_R - F\hat{x}_B^{(k)}$$

$$D_B \hat{x}_B^{(k+1)} = \hat{b}_B - E\hat{x}_R^{(k+1)}$$

in which the decoupling of the red subvector $\hat{x}_R^{(k+1)}$ and the black subvector $\hat{x}_B^{(k+1)}$ becomes obvious.