Friedrich-Alexander-Universität
Erlangen-Nürnberg

Exercise 2

Simulation and Scientific Computing I

VISHAL BODDU
SANJAY CHAMOLI
AKSHAY KAMATH

Erlangen, November 24, 2014

CONTENTS

1 Matrix A

The diagonal of the discretization matrix A is constant with a value of $4 + k^2$, on either sides of which are -1. Since we are dealing with two dimensional domain and we are following row major format to represent our unknowns $u_{i,j}$, we obtain other -1 parallel to the diagonal.

The 5 point stencil is now split, 3 of out of 5 points are -1, $4 + k^2$ and -1 which are the coefficients of $u_{i-1,j}$, $u_{i,j}$, and $u_{i+1,j}$. The other two are from a distance of Nx stride away from $4 + k^2$ towards left or towards right. In this representation we have successfully sent the known boundary terms to the right hand side.

In all the rows, since the diagonal term $4 + k^2$ is greater than $\Sigma_{n=1}^{4} 1 = 4$, we have a strictly diagonally dominant matrix (note: for top and the bottom most rows absolute sum of non-diagonal elements is just 3).

4 + k^2	-1	0	.	0	-1	0	.	.	0
-1	4 + k^2	-1	0	.	0	-1	0	.	.
0	-1	4 + k^2	-1	0	.	0	-1	0	.
.	.	.	.	.	.	.	.	.	0
0	.	.	.	.	.	.	.	0	-1
-1	0	.	.	.	.	.	.	.	0
0	-1	0	.	.	.	.	.	0	.
.	.	.	.	.	0	-1	4 + k^2	-1	0
.	.	0	-1	0	.	0	-1	4 + k^2	-1
0	.		0	-1	0	.	0	-1	4 + k^2

Table 1.1: Discretization Matrix A

2 JACOBI CONVERGENCE

In this section we will show that the Jacobi iteration scheme is convergent for given two dimensional elliptic problem. In fact even without the added term $k^2 u_{i,j}$, which in hindsight results in increase in convergence factor as the jacobi iteration scheme itself is convergent for the 2 dimensional normal poisson equation.

Assume that after discretization, we obtain the following LSE.

$$Au = b$$
$$A = D - R$$
$$u^{n+1} = D^{-1}Ru^n + D^{-1}b$$

Define $G = D^{-1}R$. We are interested in the spectral radius of G. If the spectral radius, the maximum of absolute eigenvalues of G is less than one then the iteration scheme converges.

The discretization matrix A is obtained from tensor product of two one dimensional discretization matrices A_x and A_y. Consider A_x, it is a tridiagonal matrix with $2 + \frac{k^2}{2}$ as its diagonal elements and -1 on either sides of the diagonal. Eigenvalues of A_x are

$(2 + \frac{k^2}{2} - 2cos\frac{\pi m}{N_x+1})$ where $1 \leqslant m \leqslant N_x$

The contribution of $2 + \frac{k^2}{2}$ in the eigenvalue is because of the presence of $2 + \frac{k^2}{2}$ in the diagonal. Similarly, the eigenvalues of A_y are

$(2 + \frac{k^2}{2} - 2cos\frac{\pi n}{N_y+1})$ where $1 \leqslant n \leqslant N_y$

Since $A = A_x \otimes A_y$, this implies $\lambda_A = \lambda_{Ax} + \lambda_{Ay}$ where λ denote the eigenvalues of respective matrices.

$\Rightarrow \lambda_A = (4 + k^2 - 2cos\frac{\pi m}{N_x+1} - 2cos\frac{\pi n}{N_y+1})$
Again the contribution of $4 + k^2$ is due to the diagonal D and the eigenvalues of R are $-cos\frac{\pi m}{N_x+1} - cos\frac{\pi n}{N_y+1}$

Hence, we can conclude that the eigenvalues of G are

$$\frac{-2cos\frac{\pi m}{N_x+1} - 2cos\frac{\pi n}{N_y+1}}{4 + k^2}$$

Due to the limits of m, n $cosine$ function will not take the vale of 1 or -1. This implies that the maximum absolute values of eigenvalues of G, in other words, it's spectral radius or convergence factor of Jacobi iteration scheme is

$$\frac{2cos\frac{\pi}{N_x+1} + 2cos\frac{\pi}{N_y+1}}{4 + k^2}$$

where N_x and N_y are number of nodes along X and Y direction.

Since the value of this expression is less than 1, Jacobi iterations scheme converges and we have shown a suitable boundary to the convergence factor.

$\blacksquare$

3 Performance Measurement

We see that for smaller number of nodes 32,33, the speed up reduces of we take higher number of nodes. Possible reason for this is cost of overheads for pragma openmp directives. For small problem size it might not make sense to use many CPU threads, also there is an implicit barrier in the code during computation of red black Gauss-Seidel iterations. And therefore for a small problem size a serial code should be more efficient.

We observe a saturating speed up in the case of 1024, 1025. Meaning that the gain in performance isn't much when we move from 8 to 32 threads. However we see a linear speed up upto 8 threads.

When we increase the problem size to 2048, 2049 very good speed up is obtained. Each thread is operating on one chunk of rows of u. The code appears to be scalable as performance gain does not saturate for 32 threads. In fact we appear to obtain a superlinear speed up. This is probably due to utilization caches of other cores as well in contrast to cahces of associated with single thread coupled with low cache hits of the serial code.

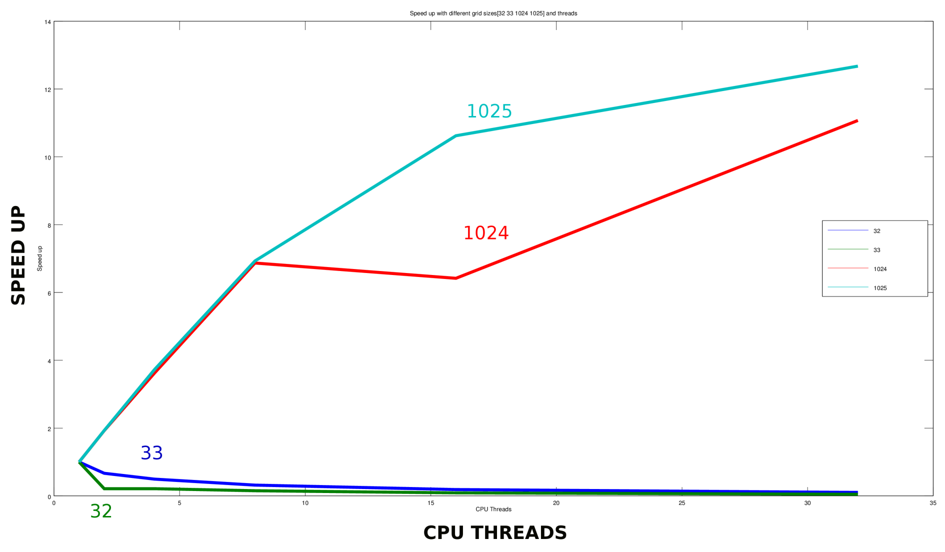


Figure 3.1: Performance Speedup Vs Number of Threads for [32 33 1024 1025] grid sizes

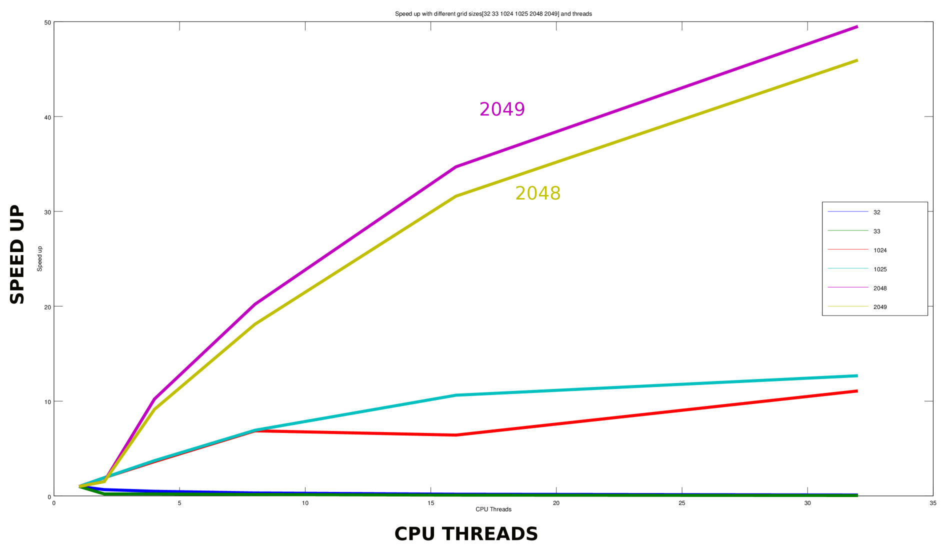
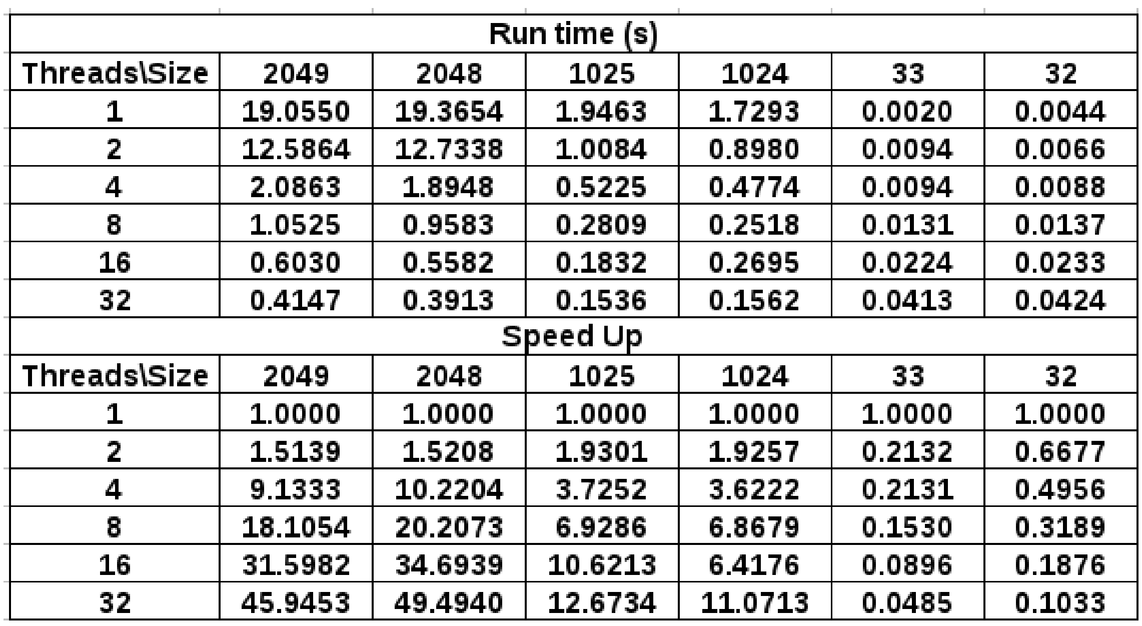


Figure 3.2: Performance Speedup Vs Number of Threads for [32 33 1024 1025 2048 2049] grid sizes



Run time (s)						
Threads\Size	2049	2048	1025	1024	33	32
1	19.0550	19.3654	1.9463	1.7293	0.0020	0.0044
2	12.5864	12.7338	1.0084	0.8980	0.0094	0.0066
4	2.0863	1.8948	0.5225	0.4774	0.0094	0.0088
8	1.0525	0.9583	0.2809	0.2518	0.0131	0.0137
16	0.6030	0.5582	0.1832	0.2695	0.0224	0.0233
32	0.4147	0.3913	0.1536	0.1562	0.0413	0.0424
Speed Up						
Threads\Size	2049	2048	1025	1024	33	32
1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2	1.5139	1.5208	1.9301	1.9257	0.2132	0.6677
4	9.1333	10.2204	3.7252	3.6222	0.2131	0.4956
8	18.1054	20.2073	6.9286	6.8679	0.1530	0.3189
16	31.5982	34.6939	10.6213	6.4176	0.0896	0.1876
32	45.9453	49.4940	12.6734	11.0713	0.0485	0.1033

Figure 3.3: Performance Measurement data