

Homework #4**Due Date: May 27, 2005****Sparse Matrix Parallel Conjugate Gradient Method****Objective:**

The objective of this assignment is to write a Poisson solver using a parallel implementation of a sparse conjugate gradient method and to investigate its efficiency and parallel scalability. For simplicity, parallelization will be done using OpenMP. Please answer ALL of the questions posed below.

Problem Definition:

The equation

$$\nabla^2 f(x, y) = k \text{ on } \Omega \quad (1)$$

is to be solved in the domain of interest $\Omega = [0, 1] \times [0, 1]$ on an evenly spaced Cartesian mesh with $n_{grid} \times n_{grid}$ mesh points. This problem is subject to the Dirichlet boundary condition that $f(x, y) = 0$ along the boundary of the domain, $\partial\Omega$. Using a second-order accurate discretization of Equation 1 we have

$$\frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\Delta x^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{\Delta y^2} = k_{i,j}. \quad (2)$$

Since $\Delta x = \Delta y = \Delta$ we can reorganize this equation to read

$$f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4f_{i,j} = k\Delta^2, \quad (3)$$

which, for the vector of unknowns $x = f_{i,j}$ can be cast in the form

$$Ax = b, \quad (4)$$

where A is a symmetric positive definite matrix (SPD) which is amenable to Conjugate Gradient (CG) solution procedures.

The matrix representation of the discrete form of Equation 3 is sparse with a pentadiagonal form. However, in this assignment you will not assume anything particular about the pattern of sparsity in the matrix, A .

Serial Implementation:

Using a sparse representation of the matrix (choose your favorite one) write a serial implementation of the conjugate gradient method for the solution of Equation 4 for

- $k = 1.0$
- $n_{grid} = 256$
- $\epsilon = 10^{-12}$

where ϵ is the convergence tolerance (the \mathcal{L}^2 -norm of the change in the residual that one must achieve through iterations before the algorithm stops.) Make sure that your executable has been compiled with optimization turned on and try to make your serial code as efficient as possible. Answer the following questions:

1. How many iterations does it take for the \mathcal{L}^2 -norm of the change in residual to achieve the desired level?
2. After this number of iterations, what is the value of $f(128, 128)$?
3. How much time (in seconds) does it take for your serial code to achieve this result? What machine did you run it on?

Parallel Implementation:

Using OpenMP, parallelize your serial implementation of the Poisson solver by parallelizing the main loops *within* the body of the CG procedure *only* (you can do more if you'd like, but this is all that is required.) Make sure that the parallel implementation computes the exact same solution as your serial implementation.

In your writeup, provide the following information:

1. Execution time, scalability, and efficiency plots using $n_{threads} = 1, 2, 4, 8, 16$. In the plot for the execution time, mark clearly the time required by the serial implementation if it is any different than that for $n_{threads} = 1$.
2. For all cases, verify and show that the value of $f(128, 128)$ is constant and correct.
3. Discuss in sufficient detail the performance of your parallel implementation with particular attention to the points of the program that present bottlenecks for parallel scalability.

Effect of Anisotropy:

Repeat the results of both the serial and parallel codes for the case where

- $k = 1.0$
- $n_{grid_x} = 128$
- $n_{grid_y} = 512$
- $\epsilon = 10^{-12}$

so that the cells in the domain have an aspect ratio of 4. What differences do you observe in the solution and in the convergence rate? Do you think this can become a problem in the physical sciences? Make sure to monitor the value of $f(64, 256)$ for this case, instead of $f(128, 128)$, which would fall on the boundary (and therefore is uninteresting).