

**CSCI-UA.0480-003**  
**Parallel Computing**  
**Lab Assignment 1**

In this lab you will implement a method for solving a group of linear equations using MPI.

**What will your program do?**

Given a set of  $n$  equations with  $n$  unknowns ( $x_1$  to  $x_n$ ), your program will calculate the values of  $x_1$  to  $x_n$  within an error margin of  $e\%$ .

The format of the file is:

- line1: #unknowns
- line2: absolute relative error
- Initial values for each unknown
- line 3 till end: the coefficients for each equation. Each equation on a line. On the same line and after all the coefficients you will find the constant of the corresponding equation.

For example, if we want to solve a system of 3 linear equations, you can have a file like this one:

```
3
0.01
2 3 4
5 1 3 6
3 7 2 8
3 6 9 6
```

The above file corresponds to the following set of equations:

$$\begin{aligned} 5X_1 + X_2 + 3X_3 &= 6 \\ 3X_1 + 7X_2 + 2X_3 &= 8 \\ 3X_1 + 6X_2 + 9X_3 &= 6 \end{aligned}$$

The third line in the file tells us that the initial values for  $X_1$  is 2, for  $X_2$  is 3, and for  $X_3$  is 4. Those values may not be the solution, or are very far from the solution that must be within 1% of the real values (as given by the 0.01 in line 2).

## How will your program do that?

We start with a set of n equations and n unknowns, like this:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nn}x_n &= b_n \end{aligned}$$

You are given all  $a_{ij}$  and  $b_1$  to  $b_n$ . You need to calculate all Xs.

Here are the steps:

1. Rewrite each equation such has the left-hand-side is one of the unknowns.

Rewriting each equation

$$x_1 = \frac{c_1 - a_{12}x_2 - a_{13}x_3 \dots - a_{1n}x_n}{a_{11}} \quad \leftarrow \text{From Equation 1}$$

$$x_2 = \frac{c_2 - a_{21}x_1 - a_{23}x_3 \dots - a_{2n}x_n}{a_{22}} \quad \leftarrow \text{From equation 2}$$

$\vdots$

$$x_{n-1} = \frac{c_{n-1} - a_{n-1,1}x_1 - a_{n-1,2}x_2 \dots - a_{n-1,n-2}x_{n-2} - a_{n-1,n}x_n}{a_{n-1,n-1}} \quad \leftarrow \text{From equation n-1}$$

$$x_n = \frac{c_n - a_{n1}x_1 - a_{n2}x_2 - \dots - a_{nn-1}x_{n-1}}{a_{nn}} \quad \leftarrow \text{From equation n}$$

Note: The Cs above refer to the constants, which are the  $b_1$  to  $b_n$ .

In general:

$$x_i = \frac{c_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}x_j}{a_{ii}}, i = 1, 2, \dots, n.$$

2. Remember that you were given some initial values for the Xs in the input file. The absolute relative error is:

$$\left| \epsilon_a \right|_i = \left| \frac{x_i^{new} - x_i^{old}}{x_i^{new}} \right| \times 100$$

Therefore, our goal is to reduce absolute relative error for each unknown to make it less or equal to relative error given in the input file (2nd line of the input file).

Note: You need to multiply the error given in the file by 100 to match it with the above equation, or to not multiply the above equation by 100.

3. Substitute the initial values in the equation of each  $X_i$  to get new value for  $X_i$ . Now we have a new set of  $X$ s.
4. Calculate the absolute relative errors for each  $X$ .
5. If all errors are equal or less the given number (2nd line in the file) then you are done.
6. Otherwise go back to step 3 with the set of new  $X$ s as  $X_{old}$ .

### What is the input to your program?

The input to your program is a text file named `xxxx.txt` where `xxxx` can be any name. We already discussed the file format.

So, if your program is called `gs` then I must be able to run your program as

```
mpirun -n x gs inputfile.txt
```

(`x` is the number of processes).

### What is the output of your program?

Your program must output to stdout (the screen) the value of each unknown. The output must look like:

```
2
3
4
```

Where 2 correspond to the value of  $X_1$ , 3 corresponds to  $X_2$ , and 4 corresponds to  $X_3, \dots$ .

In the last line of the output show the number of iterations as:

*total number of iterations: 5*

### What do I do after I finish my program?

We have provided you with a reference program *gsref* so you can check the correctness of your code. We will test your submission against this reference (for correctness not speed), but with other input files than the distributed ones.

After you finish the parallel version of your program, compile it with:

```
mpicc -o gs gs.c
```

Where `gs.c` is your source code. We provide a skeleton file to help you start.

After you compile your program and check its correctness do the following:

- Measure the time of your program (using *time* command) for 1, 2, 4, 8, 16, 32, and 64 processes. You may need to measure the time few times (~ 5) for each and take the average. Do the following with *small.txt*, *medium.txt*, *large.txt*, and *huge.txt*. Note that sometimes you may have more processes than variables. So don't worry if you don't get speedup (or even slowdown) with large number of processes. However, your code must not crash if executed with larger number of processes than variables.
- Draw a bar graph where the x-axis is the number of processes and the y-axis is the speedup relative to performance with 1 process. There must be 4 bars per x data point on the same graph (for files small, medium, large, and huge).
- What are your conclusions? To answer this question, you must write 1-2 lines explaining the behavior of each bar (small, medium, large, and huge) with the increasing number of processes. Of course writing something like "As we can see, the performance increases with the number of processes till ..." will not get you a very high grade because you are stating what we already see in the graph. You have to explain *why* we have that behavior. To do so, you may need to do some profiling as we have seen class (putting a barrier, measuring time of a part of code, etc.) to see what happens during executing.
- Include the graph and your conclusions in a single pdf file that has your name and titled *lab1 submission*

### What to submit?

Add the source code *gs.c* as well as the pdf file that contains your graph and conclusions to a **zip file named: lastname.firstname.zip**

Where lastname is your last name, and firstname is your first name.

### How to submit?

email the above zip file to your grader (not to me, not to the mailing list, and not to both graders!) with subject line: **lab1 submission**

### How will we grade this?

- Correctness: Your code must compile and run correctly on CIMS machines with any number of processes. → 5
- We will test few input files (some of them are not distributed) → 10
- The graph and interpretation → 5

This makes the total = 20 points

## **Penalties**

- If you do not follow the above protocol for submission and file name --> -1
- If your code does not compile, we will look at your source code; and out of the total of 20 points you cannot get more than 10.
- You will lose points also if your conclusions are like "As we can see from the graphs, x increasing with y". We need your explanation of what the graphs show, not your description of what we already see!