EPFL

Parallel and High Performance Computing

Dr. Pablo Antolín

Series 05 - Assignment 1

March 27 2025

Conjugate gradient

1 General information

This Series will be graded and it amounts for 25% of the total grade of the course. You have to submit your results to Moodle before April 9 at 23h59 CEST.
During the exercise session on March 27 you will have the opportunity to ask questions about the exercise, however during the exercise session on April 3 no questions will be answered about this Series.

2 Problem description

The Conjugate Gradient (CG) method is an iterative algorithm designed to solve linear systems of equations of the form $Ax = b$. For the method to be applicable, the matrix A must be real, symmetric, and positive definite, meaning that it belongs to $\mathbb{R}^{n\times n}$ with $n > 0$, satisfies $A^\top = A$, and ensures $x \cdot Ax > 0$ for all $x \in \mathbb{R}^n$ different from zero.
At each iteration k, the algorithm generates an approximate solution x_k, which is updated as

$$x_{k+1} = x_k + \alpha_k p_k.$$

Here, p_k represents the conjugate vector, also known as the search direction, and α_k is the step length for that iteration. The residual, defined as

$$r_k = b - Ax_k,$$

plays a crucial role in adjusting the search direction.
A key property of CG is that the search directions p_k are A-conjugate with all previous directions $\{p_i\}_{i=0}^{k-1}$, meaning they satisfy the condition:

$$p_k \cdot Ap_j = 0, \quad j = 0, \ldots, k-1.$$

A practical way to enforce this conjugacy is through the formula:

$$p_k = r_k - \sum_{i=0}^{k-1} \frac{p_i \cdot Ar_k}{p_i \cdot Ap_i} p_i.$$

The iterative process is typically terminated when the Euclidean norm of the residual falls below a prescribed tolerance ϵ, that is, when

$$\|r_k\|_2 < \epsilon.$$

At this point, the approximation is considered sufficiently accurate, and the algorithm concludes.

3 Basic sequential algorithm

Data: $A,\ b,\ x_0,\ \epsilon$
Result: x such that $A\,x \approx b$
initialization:
$r_0 := b - A\,x_0$
$p_0 := r_0$
$k := 0$
while $\|r_k\|_2 > \epsilon$ **do**
  $\alpha_k := \frac{r_k \cdot r_k}{p_k \cdot A\,p_k}$
  $x_{k+1} := x_k + \alpha_k\,p_k$
  $r_{k+1} := r_k - \alpha_k\,A\,p_k$
  $\beta_k := \frac{r_{k+1} \cdot r_{k+1}}{r_k\,r_k}$
  $p_{k+1} := r_{k+1} + \beta_k\,p_k$
  $k := k + 1$
end
$x := x_k$

4 The project

The goal of this project is to develop a parallel implementation of the CG algorithm for sparse matrices using MPI. The following tasks must be completed:

a) Profiling of the sequential code to determine the parts, which have to be parallelized

b) Estimate the sequential fraction of the code and predict the speedup and efficiency based on Amdahl's and Gustafson's laws, respectively.

c) Parallelize the code using MPI and measure the speedup and efficiency by performing strong and weak scaling experiments, respectively.

d) Compare your measurements to your prediction and explain your results.

5 The code

A sequential code is available in the repository of the exercises in the sub-folder `project/cg`. Matrices are read in matrix market format. An example of matrix is given in `lap2D_5pt_n100.mtx`. It corresponds to the matrix from a Poisson problem on a grid of 100×100. More examples can be found on https://sparse.tamu.edu/.
Before compiling or running the code, load

```
$> module load gcc openmpi openblas
```

Then, compile the code using the provided `Makefile`.
The provided code contains a serial implementation of a sparse solver `CGSolverSparse`. The sparse solver uses a sparse matrix storage called COO (for coordinates), sometimes also referred to as aij. It consists of 3 arrays: one for the Is (`irn`), one for Js (`jcn`), and one for the values (`a`). For a sparse matrix A we then have:

$$\begin{aligned} A_{\text{irn}[i],\text{jcn}[i]} &= a[i] \quad \forall i \in [0, \text{nnz}[\ , \\ \text{and } A_{\text{jcn}[i],\text{irn}[i]} &= a[i] \quad \text{if irn}[i] \neq \text{jcn}[i] \text{ and } \texttt{is\_sym}. \end{aligned}$$

Where nnz is the number of non-zero entries in the matrix, and `is_sym` is `true` if the matrix is symmetric.

6 Submitting the project

You must submit a report and your code. The report should be no longer than two pages. Make sure you provide answers to all questions above. The code must be submitted as a tarball archive containing all the necessary files to compile and run the code. Find some instructions on how to create a tarball archive below.

6.1 Creating a tarball on Mac/Linux

a) Open a terminal and navigate to your project directory.

b) Rename the project directory to `<first_name>_<last_name>_<SCIPER>`.

c) Run the following command:

```
$> tar -czvf <first_name>_<last_name>_<SCIPER>.tar.gz
↪  <first_name>_<last_name>_<SCIPER>
```

6.2 Creating a tarball on Windows

a) Go to your project folder and rename it to `<first_name>_<last_name>_<SCIPER>`.

b) Right-click your project folder and select `7-Zip > Add to archive`.

c) Choose `tar` as the archive format.

d) Click OK, then repeat the process, selecting `gzip` as the compression format.

e) Rename the final file to `<first_name>_<last_name>_<SCIPER>.tar.gz`.