Fast Iterative Solvers

Marcelo Cabral Filho
Mat. 404864
marcelo.cabral.filho@rwth-aachen.de

Multigrid Solvers
Project 3
SS 2023

Abstract

Implement a Lanczos method to approximate maximum eigenvalues of a Symmetric Positive Definite matrix A, and compare to a pure power iteration. The algorithm was implemented in Python. Some analyses and characteristics of the given problem will be further discussed here.

1 Introduction

In this project, algorithms for Power Iteration and the Lanczos Method were coded to find the maximum eigenvalue for a matrix (given in MSR format). Different settings and matrices were tested using both these algorithms and their performance was evaluated by comparing their runtimes.

1.1 Power Iteration Method

The algorithm for Power Iterations was used to find the largest eigenvalue of the symmetric matrix provided in the file 'power-test-msr.txt'. The convergence tolerance was set to $10e^{-8}$ and measured using the following:

$$residual = |\lambda^{(k)} - \lambda^{(k-1)}| \tag{1}$$

where $\lambda(k)$ denotes the eigenvalue calculated at the k^{th} iteration. The algorithm converged when a residual of $9.31e^{-9}$ was reached after 776 iterations and a final eigenvalue of $7.650603e^{6}$ after 4.13 seconds. It can be seen in Figure 1 that after starting with the set residual value of 1, the residual goes through some stabilization period and after about 130 iterations, it decreases linearly with small oscillations towards the end. Decreasing the tolerance further did not lead to a very different eigenvalue since with the current iterations, the vector q(776) for 776th iteration was very close to the eigenvector corresponding to $\lambda^{(1)}$, the largest eigenvalue. The residual against runtime plot is similar to the one against iteration count and is given in Figure 2.

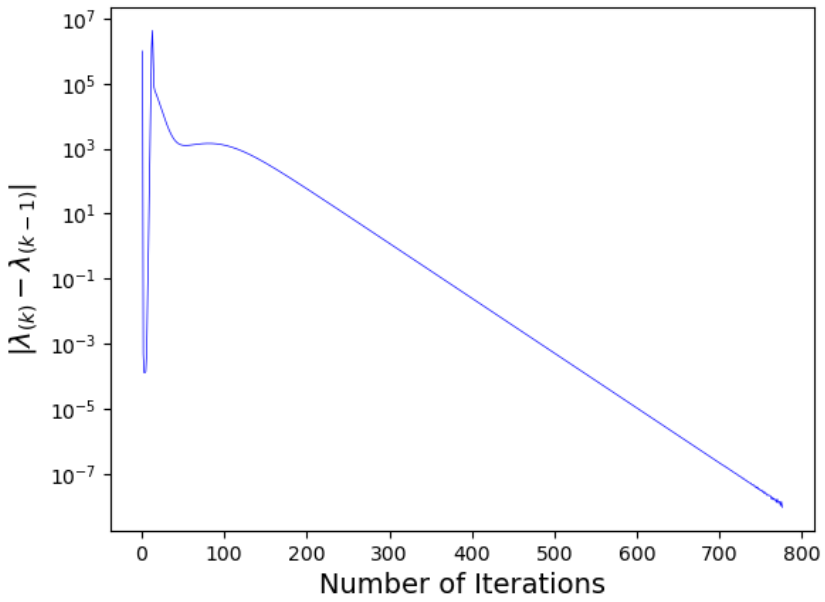

Figure 1: Residual evolution for Power Iteration algorithm - power-test-msr.txt file

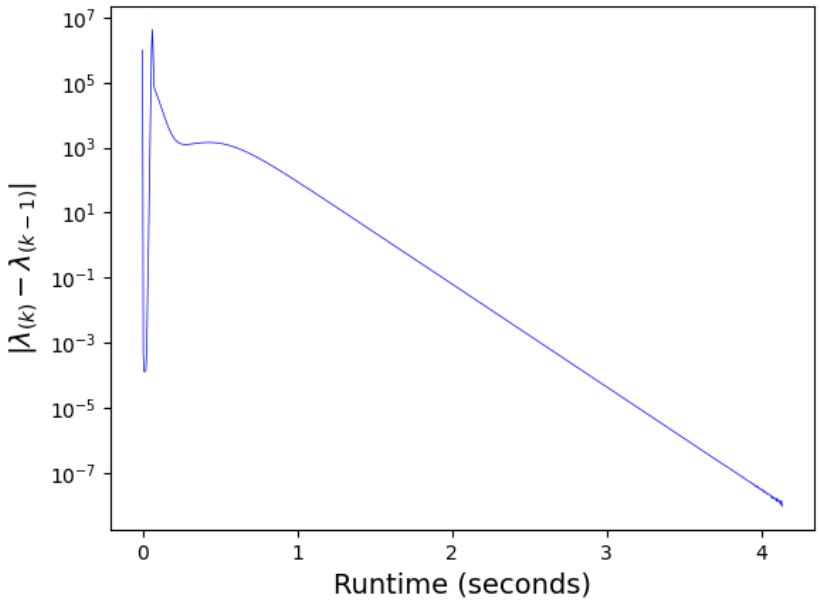

Figure 2: Residual evolution for Power Iteration algorithm over Runtime - power-test-msr.txt file

1.2 Power Iteration and Lanczos Algorithm

In this part of the project, the Lanczos algorithm was used to reduce the sparse symmetric matrix given in the file 'cg-test-msr.txt' to a tridiagonal matrix for different values of m (m = 30, m = 50, m = 75, m = 100). This matrix 'T' was then used to find an approximate maximum eigenvalue of the original matrix A using the Power Iteration algorithm. The performance of the Lanczos algorithm in conjunction with the Power Iteration algorithm was compared with Power Iteration alone in terms of iteration count and the total runtime of the programs (see Figure 3, Figure 4, Figure 5, Figure 6). From Figure 4 it is obvious that increasing the number of basis vectors leads to more iteration required to achieve the same level of convergence for the Lanczos method. It should be noted that for m greater than 50 the Power Iteration algorithm requires fewer iterations than the Lanczos method to achieve convergence criteria. However, it can be seen that for the same problem, the stand-alone Power Iteration algorithm takes substantially longer to achieve convergence when compared with the Lanczos method. This is because, with the Lanczos method, we reduce the dimensionality of the problem significantly, thereby decreasing the amount of time required to find the eigenvalue.

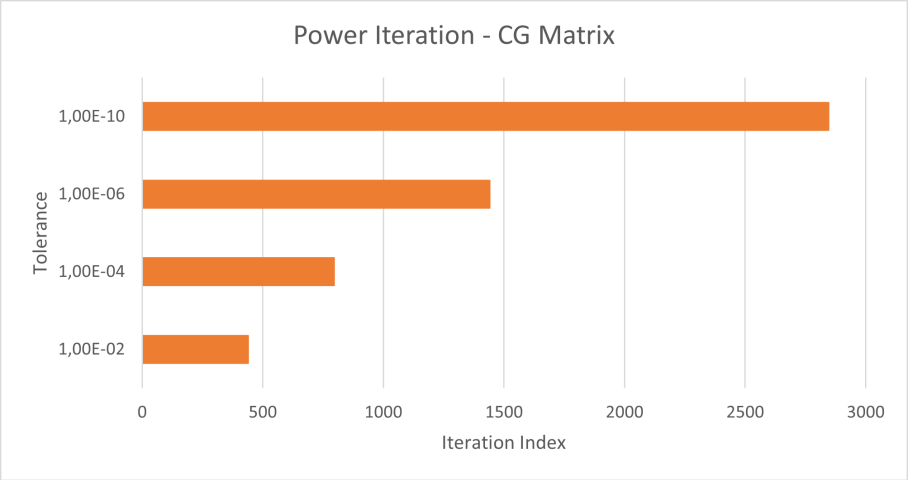

Figure 3: Overview of Power Iteration over Iteration Index

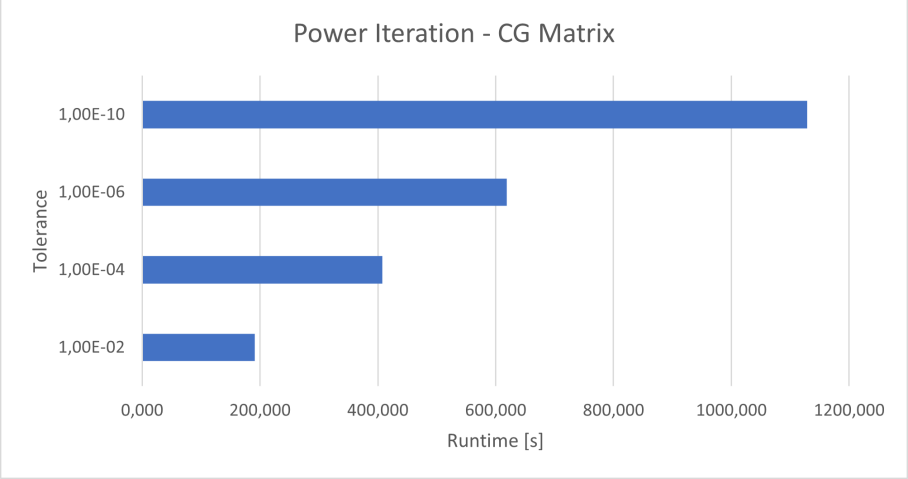

Figure 4: Overview of Power Iteration over Run Time

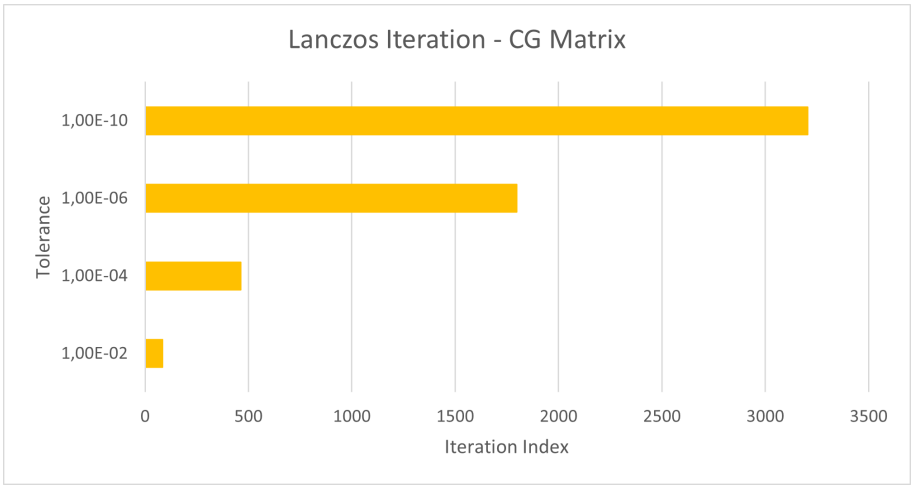

Figure 5: Overview of Lanczos Iteration over Iteration Index

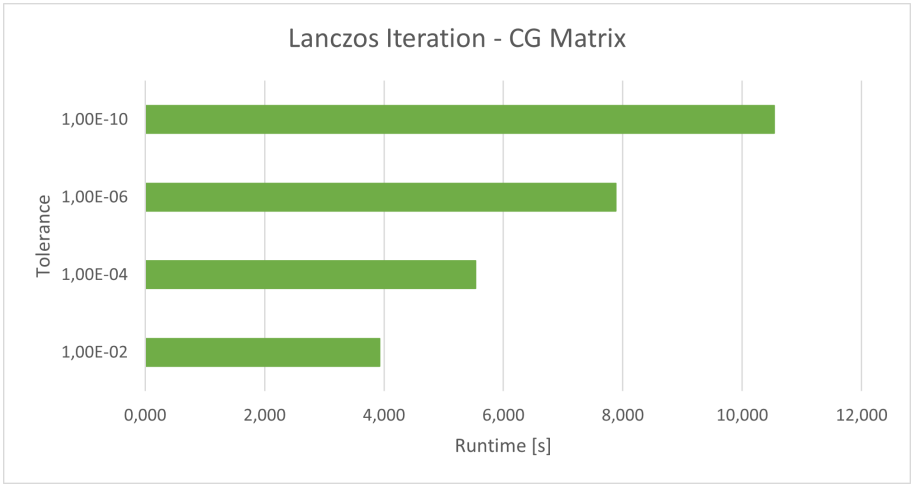

Figure 6: Overview of Lanczos Iteration over Run Time

The eigenvalues calculated from each scenario are presented in Figure 7 and Figure 8, with the exact value used to calculate the absolute difference taken to be as given in the Project 3 pdf handout. As can be seen, increasing the number of basis vectors 'm' in the Lanczos Algorithm gives a better approximation of the eigenvalue. For the case where the Power Iteration algorithm was used, it is also possible to see a better approximation of the eigenvalue but for similar results it is needed a longer run time.

Power Iteration CG Matrix					
Tol	Iterations	Final Eigenvalue	Difference	Time [Sec]	Final error
1,00E-02	440	9597,922	6,857E-01	190,705	9,96E-03
1,00E-04	796	9598,596	1,230E-02	407,644	9,95E-05
1,00E-06	1442	9598,608	1,522E-04	618,911	9,99E-07
1,00E-10	2846	9598,608	1,552E-08	1128,517	9,82E-11

Figure 7: Power Iteration Matrix Table

Lanczos Iteration CG Matrix						
Tol	m	Iterations	Final Eigenvalue	Difference	Time [Sec]	Final error
1,00E-02	30	85	9578,570	2,004E+01	3,924	9,99E-03
1,00E-04	50	464	9598,525	8,306E-02	5,532	9,86E-05
1,00E-06	75	1797	9598,608	1,521E-04	7,886	9,97E-07
1,00E-10	100	3205	9598,608	1,521E-08	10,535	9,82E-11

Figure 8: Lanczos Iteration Matrix Table