EX 3: Task 1

NSSC1

December, 2024

Group Members:

- Marcus Konrath, 01300688
- Uğur Cem Erdem, 12346958
- Katharina Perl, 01426575

Conjugate Gradients algorithm

The non-preconditioned conjugate gradients (CG) algorithm is used to solve equations in the form of

$$A \cdot x = b$$

without having to resort to the calculation of the inverse A^{-1} of the matrix A. The algorithm proves necessary when the matrix A becomes very large.

For the given scenario, we apply the CG algorithm to a sparse, symmetric, positive definite matrix. Sparse matrices are matrices that feature a relatively low proportion of non-zero entries. Symmetric matrices have their entries mirrored along the main diagonal, and thus fulfill the property

$$A^T = A \ .$$

Positive definite matrices meet the condition that, for every non-zero column vector x, the real number

$$x^T M x$$

is positive for a given symmetric matrix M.

The CG algorithm only works for symmetric, positive definite (s.p.d.) matrices.

CCS data structure

In order to minimize the memory footprint of a sparse matrix, its elements can be stored using compressed column storage (CCS) format. Instead of storing every single element, CCS only stores the non-zero values together with their corresponding row indices as well as a pointer indicating column offsets. An example is provided in Figure 1.

Since we are dealing with a symmetric matrix, the memory footprint can be even further reduced by only storing the lower triangular part of the matrix.

BCSSTK13.mtx

The matrix to be used for the CG algorithm was downloaded from an online repository called ”Matrix Market”. The matrix's identifying name is BCSSTK13. It measures 2003 x 2003 and contains 83883 non-zero elements, which makes it a sparse matrix with roughly 2.1 % of non-zero entries. Due to the symmetry

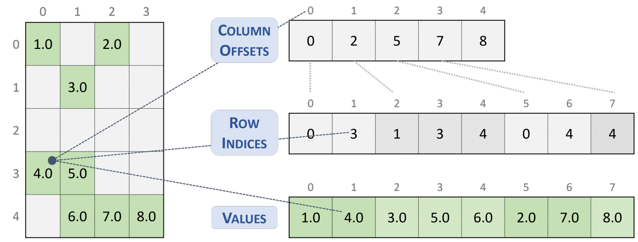

Figure 1: Example of CCS format
https://docs.nvidia.com/cuda/cusparse/index.html#csr-format

of the matrix, only the lower triangular part is actually stored in the file BCSSTK13.mtx (which holds 42943 entries).

The structure plot of the matrix is shown in Figure 2.

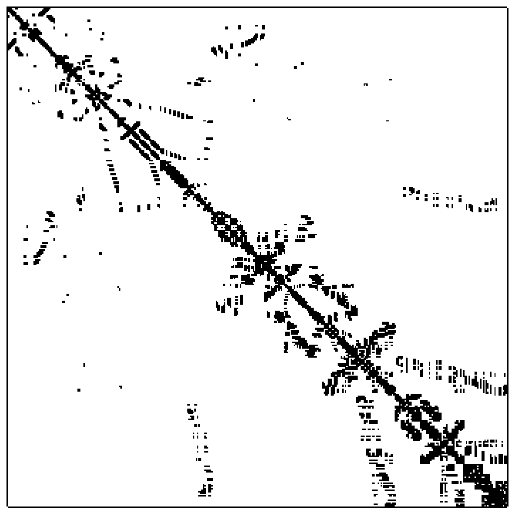

Figure 2: Structure plot of BCSSTK13
https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/bcsstruc1/bcsstk13_lg.html

CG algorithm (continued)

The following indicators of the algorithm's performance were calculated:

- $\frac{||r_k||_2}{||r_0||_2}$ (residual norm)
- $||e_k|| = ||x^* - x_k||_A = \sqrt{e_k^T A e_k}$ (error on the A-norm)

Figures 3 and 4 show the plots of these indicators against the number of iterations. The maximum number of iterations was set to 100.

Figure 3 indicates a steep increase in the algorithm's accuracy (or, a steep decrease in the value of the residual norm) within the first 10 iterations. After about the 20^{th} iteration, the gain in accuracy gradually decreases. The increase in accuracy (or, the decrease in the value of the residual norm) indicates that the algorithm is converging towards the actual solution of the equation to be solved.

Figure 4 shows a very similar plot, but a glance at the scale of the y-axis reveals that the values of the A-norm are very high. Such high values could indicate that the matrix is ill-conditioned when it comes to calculating the inverse A^{-1}, meaning that small perturbations in the right-hand-side vector b result in large changes in the result vector x. It is therefore justified to use the CG algorithm in this case to solve the linear equation $Ax = b$. After all, just like in Figure 3, the values converge towards zero, indicating that the algorithm is working as expected.

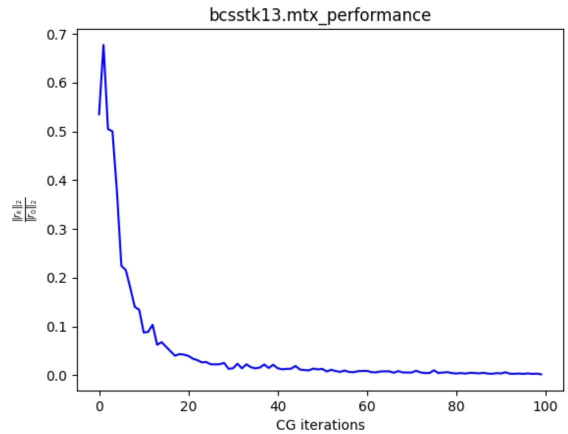

Figure 3: Assessing CG performance using residuals

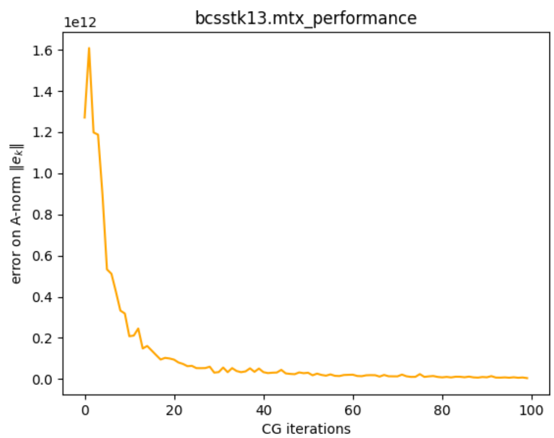

Figure 4: Assessing CG performance using error on A-norm

We can also see that for both performance indicators, their initial values are relatively high, which is due to the initial guess made that is necessary for the CG algorithm to function. In our case, the initial guess was $x^* = [0, \ldots, 0]$, while the expected result vector was $x = [1, \ldots, 1]$.

Since the task sought for the implementation of a non-preconditioned version of the CG algorithm, an increase in the convergence rate can be expected when a preconditioner is included in the implementation of the algorithm.