Computational mathematics for learning and data analysis: Project 21

Vitiello Carmine, Lipari Giuseppe

Master Degree in Computer science.
c.vitiello6@studenti.unipi.it, g.lipari@studenti.unipi.it.
Computational mathematics for learning and data analysis, Academic Year: 2022/2023

November 9, 2023

Abstract

In this report, we implement and assess project #21 assigned in the Computational Mathematics for Learning and Data Analysis course. The aim of the project is the resolution of a Min-Cost-Flow problem using the Conjugate Gradient method and one between the Minimal Residual Method (MINRES) and the Generalized Minimal Residual method (GMRES).

1 Introduction

The problem is a sparse linear system on the form:

$$\begin{bmatrix} D & E^T \\ E & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}$$

Where $D \in \mathbb{R}^{mxm}$ is a diagonal positive definite matrix and $E \in \mathbb{R}^{nxm}$ is the node-arc incidence matrix of a given directed graph.
These problems arise as the KKT system of the convex quadratic separable Min-Cost-Flow Problem. We used for the problem instance generation random generators available at this link

(A1) is Conjugate Gradient for linear system applied to the "reduced" linear system

$$(ED^{-1}E^T)y = ED^{-1}b - c.$$

(Indeed, one can see that eliminating x the two linear systems are equivalent). The implementation hasn't required the matrix $ED^{-1}E$ explicitly but relied on a callback function that computed the product $ED^{-1}Ev$ for a given vector v.

(A2) is GMRES (or MINRES), applied to the original system.
No off-the-shelf solvers were used.

1.1 Karush–Kuhn–Tucker conditions (KKT)

KKT defines the necessary condition for a nonlinear problem with constraints to get a solution. Given general problem

$$\begin{aligned} &min\ f(x) \\ &subject\ to \quad g_i(x) \leq 0, i \in \mathbb{I} \\ &\qquad\qquad h_j(x) = 0, j \in \mathbb{J} \end{aligned}$$

where $x \in X$ is the optimization variable chosen from a convex subset of $\mathbb{R}^n$, f is the objective function, g_i elements are the inequality constrain functions and h_j are the equality constrain functions. The constraints delimit the set of feasible solutions X. The Karush-Kuhn-Tucker optimality conditions (KKT conditions) are defined from:

$\exists \lambda \in \mathbb{R}_+^{\#\mathbb{I}}$ and $\mu \in \mathbb{R}^{\#\mathbb{J}}$

- Primal feasibility: $g_i(x) \leq 0, h_j(x) = 0 \quad \forall\ i, j$ (KKT-F)
- Stationarity: $\nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x) + \sum_{j=1}^{r} \mu_j \nabla h_j(x) = 0$ (KKT-G)
- Complementary Slackness: $\lambda_i g_i(x) = 0 \quad \forall\ i \in \mathbb{I}$

1.2 Min Cost Flow Problem

Similarly, as reported for the multi-commodity min-cost network-flow problem in chapter 2.3 of [1], we can describe the Min Cost Flow problem as follows: Given a directed graph $G(N, A)$ with a finite set of nodes $n \in N$ and a finite set of arcs $a \in A$. Each arc has the following:

- two key attributes, namely its tail $t(a) \in N$ and its head $h(a) \in N$
- max capacity c_a (the flow upper bound of the arc)
- cost w_a per unit flow

We have twice further data:

- b_n is the deficit or surplus flow of the node n
- x_a is the amount of the flow in the arc a from its head to its tail

The objective is to find a minimum-cost total flow that is non-negative and respects the flow upper bounds on the arcs.
We can formulate this as follows:

- $min \quad \sum_{a \in A} w_a x_a$ (Total cost of the flow)
- $0 \leq x_a \leq c_a, \qquad \forall a \in A$
- $\sum_{a \in A | t(a) = u} x_a - \sum_{a \in A | h(a) = u} x_a = b_n, \qquad \forall n \in N$

1.3 Reduced Linear system

Starting from the original linear system, we can reduce it to obtain the given linear system: $(ED^{-1}E^T)y = ED^{-1}b - c$

$$\begin{cases} Dx + E^T y = b \\ c = Ex \end{cases} = \begin{cases} x = -D^{-1}E^T y + D^{-1}b \\ c = Ex \end{cases}$$

$$c = E(-D^{-1}E^T y + D^{-1}b) = -ED^{-1}E^T y + ED^{-1}b$$
$$(ED^{-1}E^T)y = ED^{-1}b - c$$

1.4 Incidence Matrix

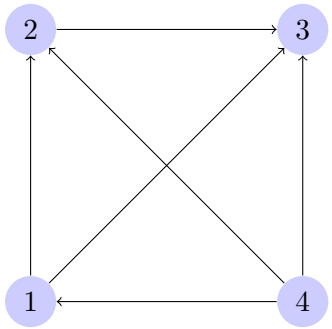
An incidence matrix is a logical structure that shows the relationship between the nodes and arcs (commonly called vertices and edges) in a graph. A matrix row represents a node i, instead, the columns show the arcs connected with the node i. The incidence matrix can have three types of value in a cell (i, j):

$$E_{ij} = \begin{cases} -1 & \text{if edge } e_j \text{ leaves vertex } v_i, \\ 1 & \text{if edge } e_j \text{ enters vertex } v_i, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

For example, if the $i-th$ row has these values [0 1 0 -1] then the second arc goes in the i-th node and the fourth arc goes out. We have a directed graph, so the incidence matrix is $E \in \mathbb{R}^{nxm}$ where n and m are the number of nodes and arc.

During the problem study, we used some generators that take input parameters for each problem instance and write a file *.dmx with the relative information of the graph topology. We developed a Python script that reads the different topologies graph's files and parses them to be used as inputs for the algorithms. We can build the E Incidence Matrix through these graphs' inputs. One possible incidence matrix could be like this:

$$E = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \\ n_4 \end{array} \begin{array}{c} \begin{array}{cccccc} (1-3) & (1-2) & (2-3) & (4-1) & (4-2) & (4-3) \end{array} \\ \begin{bmatrix} -1 & -1 & 0 & +1 & 0 & 0 \\ 0 & +1 & -1 & 0 & +1 & 0 \\ +1 & 0 & +1 & 0 & 0 & +1 \\ 0 & 0 & 0 & -1 & -1 & 1 \end{bmatrix} \end{array}$$



1.4.1 Parsing Algorithm

Each generated line of the graph file has a specific format to show clearly all information. For example, for each arc of the graph, there is a line like this: a 1 2 0 8 6.
The first letter indicates an arc; the first two numbers are the source(1) and destination(2). The last couple shown with the values 8 and 6 are the max capacity and the cost. We want to focus on creating arcs because the making of the incidence matrix is based on them.

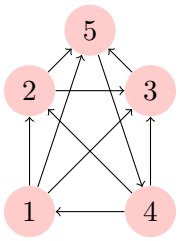
Algorithm 1 Generated Graph Parsing

```
File F <- readFile()
new Matrix M
indexArc <- 0
for i in F do    ▷ in F means that we read all lines of file per iteration
if( fist Letter of i = 'a') then
   new Arc a
   a.index <- indexArc
   indexArc <- indexArc + 1
   a <- cost,Max capacity,source and destination
   M[source][a.index] <- 1
   M[destination][a.index] <- −1
end for
```

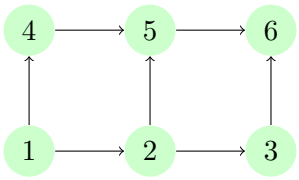
1.5 Topology of the Node Graphs

For the experiment, we used some different topologies of graphs, in fact, sometimes the results are different for the specific topology of the network. We used three topologies which are:

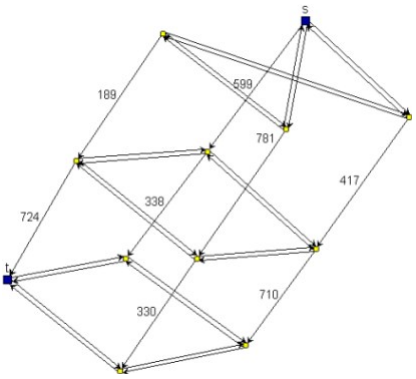
- Complete Graph: It is a graph that has n nodes that are connected with all the others that are included in the network.



- Grid Graph: It has a regular structure like a Grid, and this topology is used for projects with many devices or sensors.

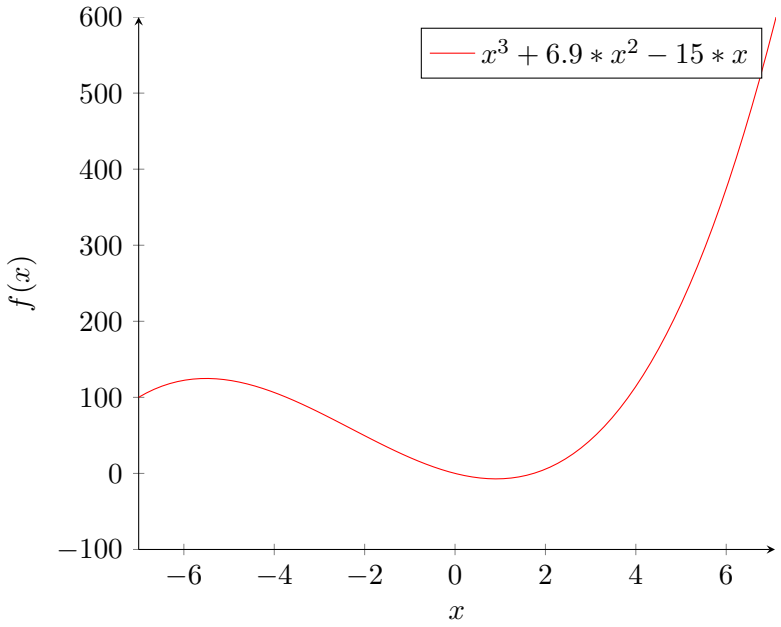


- RMF Graph: It is a particular network composed of some structures, called frames, with base, height, and volume. We can usually compose the frames to create a complex network.



2 Methods: Conjugate Gradient for linear system

The conjugate gradient (CG) is an iteration method presented in the book [2] in the 1950s for solving linear systems like $Ax = b$, associated with the numerical solutions of linear partial differential equations.
The algorithm can find the global solution if the matrix A is positive, definite and symmetric. Because the method changes the direction depending on the gradient $\nabla f(x)$ until the gradient does not tend to zero. If the matrix has these properties, then the gradient computing is easier, and it can find a global solution. For example, in the above graph, the point near $x = -4$ is not a global minimum, but it's local, so we have to avoid them.



We must verify if the matrix $ED^{-1}E^T$ satisfies the preconditions. We can compute a generic example matrix to analyze the symmetry.

$$E \in \mathbb{R}^{nxm} \; E = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \end{bmatrix}$$

$$D \in \mathbb{R}^{mxm} \text{ and } \forall i \; d_i > 0 \; D = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix}$$

$$ED^{-1}E^T = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \end{bmatrix} \begin{bmatrix} \frac{1}{d_1} & 0 & 0 \\ 0 & \frac{1}{d_2} & 0 \\ 0 & 0 & \frac{1}{d_3} \end{bmatrix} \begin{bmatrix} e_1 & e_4 \\ e_2 & e_5 \\ e_3 & e_6 \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{e_1}{d_1} & \frac{e_2}{d_2} & \frac{e_3}{d_3} \\ \frac{e_4}{d_1} & \frac{e_5}{d_2} & \frac{e_6}{d_3} \end{bmatrix} \begin{bmatrix} e_1 & e_4 \\ e_2 & e_5 \\ e_3 & e_6 \end{bmatrix} = \begin{bmatrix} \frac{e_1^2}{d_1} + \frac{e_2^2}{d_2} + \frac{e_3^2}{d_3} & \frac{e_1 e_4}{d_1} + \frac{e_2 e_5}{d_2} + \frac{e_3 e_6}{d_3} \\ \frac{e_1 e_4}{d_1} + \frac{e_2 e_5}{d_2} + \frac{e_3 e_6}{d_3} & \frac{e_4^2}{d_1} + \frac{e_5^2}{d_2} + \frac{e_6^2}{d_3} \end{bmatrix}$$

With this example, we get an insight into the symmetry property's proof, but we have not proven it yet. To formalize the proof, we have to take its definition. The matrix is symmetric if only if $A = A^T$, then we must compute A^T to see if the transposed equals A.

$$ED^{-1}E^T = (ED^{-1}E^T)^T = (E^T)^T(D^{-1})^T E^T = ED^{-1}E^T$$

D^{-1} is equal to $(D^{-1})^T$ because it is symmetric, and then the relation is proved. Instead, the positive definite property can not be proved on the whole matrix, as shown in a theorem which appears in [3] about the incidence matrix rank, that sets out a graph $G(N, A)$ connected and oriented has incidence matrix $E \in \mathbb{R}^{nxm}$ with rank equal to $n - 1$.

The first cause is that all the columns have $+1$ and -1 elements, and the rows are not linear independent because their sum always gives results equal to zero. This fact shows that the rank of the incidence matrix will not be full, but it can be $\leq n - 1$, so we want to know its rank.

We have to compose a relation such that the rank of E has to be $\geq n-1$; thus, we can set out that it will be exactly equal to $n - 1$.

The proof is worded by the definition of the connected graph $G(N, A)$ because the minimum number m of the arcs has to be $m \geq n - 1$, then we can't delete more than one arc to have an incidence matrix with rows linear independents. Otherwise, we contradict the start conditions of the problem because we will have an unconnected nodes graph.

To show the proof that the rank of A is equal to $n-1$, we want to analyze the singular product of this matrix

$$A = ED^{-1/2}D^{-1/2}E^T$$

. We can show that $ED^{-1/2}$ and $D^{-1/2}E^T$ have the same rank, which is equal to $n-1$ because there is the statement of a theorem of the linear algebra that proves them. If we have a product between matrix C and an invertible matrix F, then the

$$rank(C) = rank(CF) = rank(FC)$$

D is an invertible positive diagonal matrix, then the relation is valid for this case.

$$rank(E) = rank(ED^{-1/2})$$
$$rank(E^T) = rank(D^{-1/2}E^T)$$

Now, we can rename the previous two parts in

$$B = ED^{-1/2} \quad \text{and} \quad B^T = D^{-1/2}E^T$$

because the last theorem on the product of the matrix and his transposed says:
"If the matrix B is over the real numbers, the $rank(BB^T) = rank(B^TB) = rank(B) = rank(B^T)$ " [4], then we prove that the rank of A is equal to $n-1$.

2.1 Algorithm

We want to solve the linear system $Ax = b$ where A is a square matrix positive definite.
We use an optimization interpretation of the problem:

$$min\ \frac{1}{2}\ x^T Ax - b^T x + const = min\ \frac{1}{2}\|x - x_*\|_A^2 = min\ \frac{1}{2}(x - x_*)^T A(x - x_*)$$

with x_* exact solution.
We have to compute the residual $r(x) < -\nabla f(x) = A * x - b$, where r has to be zero like the $\nabla f(x)$.
The idea of the algorithm is to use a vector for the direction called d_k and a step α_k for each iteration i.
The value $\alpha = -\frac{\nabla f(x_k)^T d_k}{d_k^T A d_k} = -\frac{r_k^T d_k}{d_k^T A d_k}$
In the iterative method, the direction set $d_0, d_1, ..., d_k$ has to be conjugate, which means a couple $v, u \in \mathbb{R}^n$ can be mutually conjugate concerning a symmetric positive definite matrix A if u and Av are mutually orthogonal $(u, Av) = u^T Av = 0$.

Algorithm 2 Conjugate Gradient

$x = x0$
$r = b - A@x0$ ▷ residual r = b-Ax but we start with x=0 so is equal to b
$d = r$ ▷ direction
for `j in maxIteration` **do**
  `Ad=` A `@` d
  `num`$\alpha = r^T$ `@` r
  `den`$\alpha = d^T$ `@ Ad`
  $\alpha =$`num`α`/den`α
  $x = x + \alpha * d$
  $r = r - \alpha$`* Ad`
  $currentTollerance = r^T @ r$
  `if` $currentTollerance <= tollerance^2$
  $\beta = r^T$`@`r `/num`α ▷ r is the new value
  $d = r + \beta$`*d`
end for

A breakdown (division by zero in α and β computation) only happens once we achieve convergence when at j step $x_j = x_*$ and the residual $r_j = 0$

2.2 Convergence

The method converges in a finite number of steps:
$A \in \mathbb{R}^{mxm}$ for Krylov subspaces $K_n(A, b) \subseteq \mathbb{R}^m$
We cannot have more than m linearly independent vectors, so r_m must be equal to zero (unless it was already zero at a previous iteration).
Error Bound Theorem:

$$\frac{\|x_n - x_*\|_A}{\|x_0 - x_*\|_A} \leq (\frac{\sqrt{\lambda_{max}} - \sqrt{\lambda_{min}}}{\sqrt{\lambda_{max}} + \sqrt{\lambda_{min}}})^n$$

where $\lambda_{min}, \lambda_{max}$ are the smallest/largest eigenvalue of A

or in terms of condition number:

$$K(A) = \|A\| \cdot \|A^{-1}\| = \frac{\lambda_{max}}{\lambda_{min}}, \frac{\sqrt{\lambda_{max}} - \sqrt{\lambda_{min}}}{\sqrt{\lambda_{max}} + \sqrt{\lambda_{min}}} = \frac{\frac{\sqrt{\lambda_{max}}}{\sqrt{\lambda_{min}}} - 1}{\frac{\sqrt{\lambda_{max}}}{\sqrt{\lambda_{min}}} + 1} =$$

$$\frac{\sqrt{K(A)} - 1}{\sqrt{K(A)} + 1} = 1 - 2\sqrt{\frac{1}{K(A)}} + O(\frac{1}{K(A)}$$

So, if A is ill-conditioned, the method converges slowly. This bound is often sub-optimal; the true convergence speed depends on the location of eigenvalues of A. In particular, if well divided into clusters.

In this particular situation, the rank is equal to $n - 1$ and the λ_{min} is equal to zero, then there will be a problem with the denominator of the conditioning number $K(A)$, in addition, the Conjugate Gradient method has to have a matrix positive definite, namely must be full rank. This case does not make it impossible to compute the solution of the linear system $Ax = b$ because we can reduce the dimensions of the matrix A to $n - 1$ to make true the preconditions of the algorithm. The algorithm produces the best result at the k-th repetition unless the stop condition is satisfied.

2.3 Cost

The cost of the method depends on whether the problem is ill-conditioned because the conditioning value of the problem changes the minimum number of steps. We can prove that the algorithm computes the best approximation in k steps, and then the conditioning number of the problem is part of the cost value. The time cost is $O(nnz * k)$, where nnz are the elements, not zero, of the matrix A. Instead, the algorithm needs to have three vectors for the space cost. The vectors create the history of the solution $x_1, ..., x_n$, the direction $d_1, ..., d_{n-1}$ and the residual $r_1, ..., r_{n-1}$.

3 Iterative Methods: MINRES, GMRES

The Minimal Residual Method (MINRES) and the Generalized Minimal Residual Method (GMRES) are two iterative methods for the numerical solution of the linear equations system $Ax = b$. The couple is based on orthogonal projection processes onto Krylov subspaces that have the form $K_n(A, v) \equiv span\{v, Av, A^2v, \ldots, A^{n-1}v\}$ and the dimension increases by one at each step of the approximation process. The elements of subspace are composed of vectors of the form $p(A)v$, where p is a polynomial.

The GMRES uses Arnoldi's Method defined in [5] to build an orthonormal base V_m of the Krylov subspace $K_n(A, r_0)$ ($r_0 = b$ when $x_0 = 0$), and it's an efficient technique for approximating the solution of large sparse linear systems of equations.

The MINRES uses the symmetric Lanczos algorithm, which can be viewed as a simplification of Arnoldi's method when the matrix A is symmetric.

Each iteration minimizes the residual $r_n = \|b - Ax_n\|$ with $x_n, \in K_n$ and stops when they are sufficiently close to a solution. The solution x_n can be substituted by a linear composition with the orthonormal vectors of $V_n y$, then $x_n = V_n y$ with $y \in \mathbb{R}^n$ become the vector we need to find. The Arnoldi's method in the process also constructs a matrix $H_{n+1,n}$ Hessenberg matrix that becomes symmetric tridiagonal T_n (without the last row $T_{n+1,n}$) when A is symmetric. This decomposition satisfies the relation

$$AV_n = V_{n+1}H_n$$

Now, we can analyze all the steps of the relation:

$$\|r_n\| = \|b - Ax\| = \|r_0 - AV_n y\| = \|\beta_1 v_1 - V_{n+1}H_n y\| = \|V_{n+1}(\beta_1 e1 - H_n y)\| = \|\beta_! e1 - H_n y\|$$

where $e1 = (1, 0, 0, \ldots, 0)^T$ and $\beta_1 = \|r_0\|$. The new subproblem is

$$y_n = \arg\min_{n \in V_n} \|\beta e1 - H_n y_n\|$$

Both methods are valid solutions for our problem; we choose to implement the MINRES method, where only the symmetry of the matrix A is mandatory and requires less memory of GMRES because, as we will describe, it doesn't need to be kept in memory the matrix V_m.

3.1 MINRES method

The MINRES proposed in [6] is a logical development of the Lanczos algorithm for tridiagonalizing A; this approach suggests numerical algorithms for solving such systems when A is a large and sparse symmetric but indefinite. The main objective is to find an approximate solution that minimizes the residual

$$r_x = \|b - Ax\|$$

With A is symmetric, the Lanczos process transforms the Arnoldi relation $V_k^T A V_k = H_k$. It follows that H_k must also be symmetric and, therefore, tridiagonal. We will denote $T_k \equiv H_k$ and $T_{k+1,k} \equiv H_{k+1,k}$

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_k \\ & & & \beta_k & \alpha_k \\ & & & & \beta_{k+1} \end{bmatrix}$$

The Lanczos process iteratively computes vectors v_k as follows:

$$v_0 = 0, \quad \beta_1 v_1 = b, \quad \text{where } \beta_1 = \|b\| \text{ serves to normalize } v_1,$$
$$p_k = A v_k, \quad \alpha_k = v_k^T p_k,$$
$$\beta_{k+1} v_{k+1} = p_k - \alpha_k v_k - \beta_k v_{k-1}, \quad \text{where } \beta_{k+1} \text{ serves to normalize } v_{k+1}$$

so that any Krylov basis vector v_{k+1} can be obtained using a 3-term recurrence.

Algorithm 3 Lanczos

Require: A, v_k, v_{k-1}, β_k
Ensure: $\alpha_k, \beta_{k+1}, v_{k+1}$
$p_k = A * v_k$
$\alpha_k = v_k^T * p_k$
$p_k = p_k - \alpha_k * v_k$
$v_{k+1} = p_k - \beta_k * v_{k-1}$
$\beta_{k+1} = \|v_{k+1}\|$
if $\beta_{k+1} \neq 0$ **then**
$v_{k+1} = v_{k+1}/\beta_{k+1}$
end if

In the end, the algorithm checks if the β_{k+1} value is close to zero because it could be a lucky case where the columns of V_k are orthonormal, the process stops ($k \leq n$) and $AV_k = V_k T_k$. Within each Lanczos iteration, we solve the subproblem

$$y_k = \arg\min_{y \in \mathbb{R}^k} \|\beta_1 e_1 - T_k y\|$$

by computing a QR factorization. We perform the Givens rotation to yield an upper triangular matrix to compute the QR decomposition.

$$Q_k T_k = \begin{bmatrix} \underline{R}_k \\ 0 \end{bmatrix} = \begin{bmatrix} \gamma_1 & \delta_2 & \epsilon_3 & & & \\ & \gamma_2 & \delta_3 & \epsilon_4 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \epsilon_k \\ & & & & \ddots & \delta_k \\ & & & & & \gamma_k \\ & & & & & 0 \end{bmatrix}, \qquad Q_k(\beta_1 e_1) = \begin{bmatrix} t_k \\ \phi_k \end{bmatrix}$$

where $t_k = [\tau_1, \tau_2, \ldots, \tau_k]^T$
Differently from the GMRES method, for MINRES, there is no need to save the Krylov basis generated by the Lanczos algorithm. The current x_k can be directly computed by x_{k-1}.
Let $\underline{R}_k$ the square submatrix of R_k obtained by eliminating the last row. We define $P_k = [p_1, \ldots, p_k] = V_k \underline{R}^{-1} \implies P_k \underline{R}_k = V_k$. Since $\underline{R}_k$ has only three nonzero diagonals, we have:

$$p_k = \frac{v_k - \delta_k p_{k-2} - \epsilon_k p_{k-1}}{\gamma_k}$$

Now, recalling the solution of the last squares problem using the QR factorization $y = \underline{R}^{-1} Q^T \beta_1 e_1$, we have that:

$$x_k = V_k y_k = V_k \underline{R}^{-1} \underline{Q}^T \beta_1 e_1 = P_k \begin{bmatrix} t_k - 1 \\ \tau_k \end{bmatrix} = P_{k-1} t_{k-1} + p_k \tau_k = x_{k-1} + p_k \tau_k$$

Algorithm 4 MINRES

Require: $A, b, maxiter$

$\beta_1 = \|b\|$
$r = b$
$v_0 = 0$
$v_k = b/\beta_1$
for $k \leq$ maxiter **do**
  $\alpha_k, \beta_{k+1}, v_{k+1} \leftarrow Lanczos(A, v_k, v_{k-1}, \beta_k)$
  $T \leftarrow tridiag(\beta_k, \alpha_k, \beta_{k+1})$ ▷ function that update a T matrix
  $Q, R = QR_{factorization}(T)$ ▷ factorization, we used Givens Rotations
  $\tau_k = G_{k,k}^T * \beta_1$
  if $k = 1$ **then**
    $p_k = \frac{v_k}{R[k,k]}$
  else if $k = 2$ **then**
    $p_k = \frac{v_k - p_{k-1} * R[k-1,k]}{R[k,k]}$
  else if $k \geq 3$ **then**
    $p_k = \frac{v_k - p_{k-2} * R[k-2,k] - p_{k-1} * R[k-1,k]}{R[k,k]}$
  end if
  $x = x + \tau_k * p_k$
  $r = \|b - Ax\|$
  if $r \leq tollerance$ **then**
    $return \quad k, r, x$
  end if
end for

3.2 Convergence

For positive definite matrices, the rate of convergence for the MINRES method algorithm can be estimated similarly to the CG algorithm. However, there is a distinction in that the estimation is not related to the errors of the iterative solutions but rather to the residual. The following statement is valid:

$$\|r_k\| \leq 2\left(\frac{\sqrt{k(A)}-1}{\sqrt{k(A)}+1}\right)^k \|r_0\|,$$

where $k(A)$ is the condition number of matrix A. Because A is normal, we have

$$k(A) = \frac{|\lambda_{max}(A)|}{|\lambda_{min}(A)|},$$

where $\lambda_{max}(A)$ and $\lambda_{min}(A)$ are maximal and minimal eigenvalues of A, respectively. if A has only n distinct eigenvalues, it converges in n steps.

In MINRES and GMRES, algorithms do not need a positive definite matrix; namely, having all the eigenvalues greater than zero is unnecessary. We can use the methods with max iterations equal to the rank of the matrix A because the algorithms compute the better approximation of the solution at the $k-th$ iteration than the previous repetitions, so it's not beneficial to set the max iteration number greater than the number of the rows linear independent. However, it's possible to compute the rank to get further proof that the matrix is not always full rank.

To compute the rank of the block matrix, the Schur Complement Theorem describes the relation between the blocks of the matrix and the properties as the rank of the whole matrix, determinant and others.

$$A = \begin{bmatrix} B & C \\ D & E \end{bmatrix}$$

$$rank(A) = rank(B) + rank(E - DB^{-1}C)$$

For this specific problem, the matrix is equal to:

$$A = \begin{bmatrix} D & E^T \\ E & 0 \end{bmatrix}$$

$$rank(A) = rank(D) + rank(0 - ED^{-1}E^T) = m + n - 1$$

Hence, the λ_{min} equals zero, but this is not an issue as we said before.

3.3 Cost

For GMRES, the cost of the algorithm is due to Arnoldi's method. Thus, an estimate of the cost in time can be obtained from

$$n(\text{matrix-vector}) + O(mn^2)$$

In addition, it should be noted that to compute a new solution, it is necessary to store all the m vectors of the basis.
If A is symmetric, then H_n is also "symmetric", and we can use MINRES (Lanczos iteration).

$$H_{ij} = H_{ji}\text{: follow from } H_{ij} = q_i^T A q_j = (q_i^T A q_j)^T = q_j^T A q_i = H_j i$$

So the orthogonalization loop can be reduced of the "for" cycle because we have only two vectors to check, and we already know that $H(j-1,j) = H(j,j-1)$

$$\text{Total cost: } n \text{ (matrix-vector)} + O(mn)$$

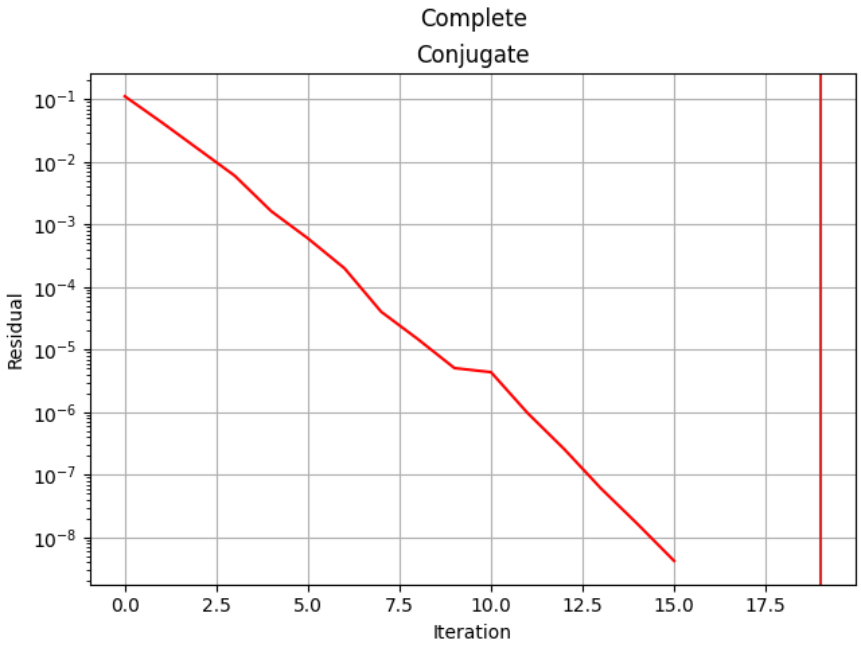
The symmetry property reduces the cost of the orthogonalization loop and the small-scale least-squares systems: H_n is tridiagonal.

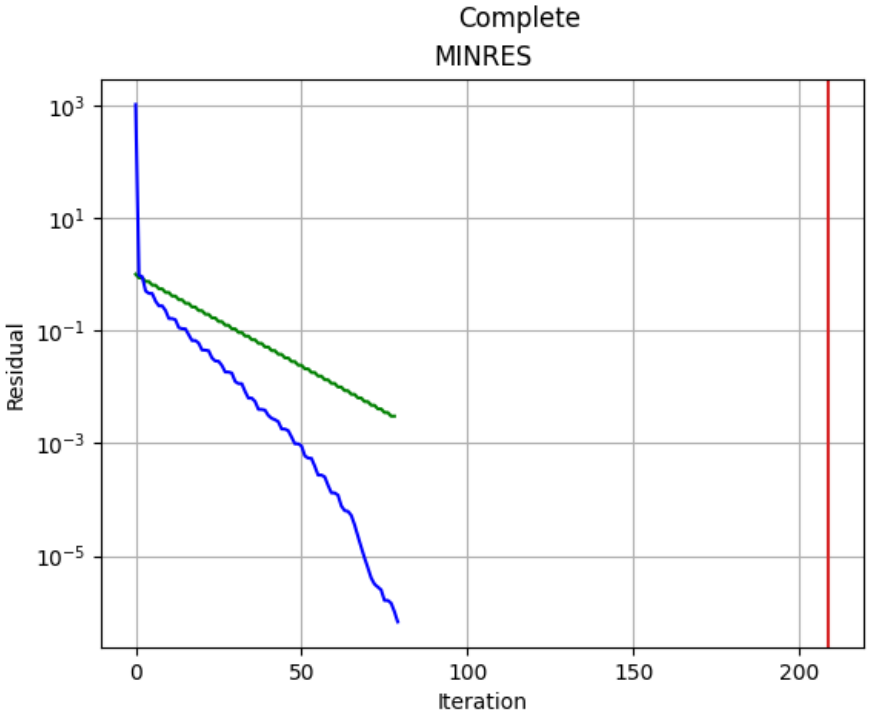
4 Results

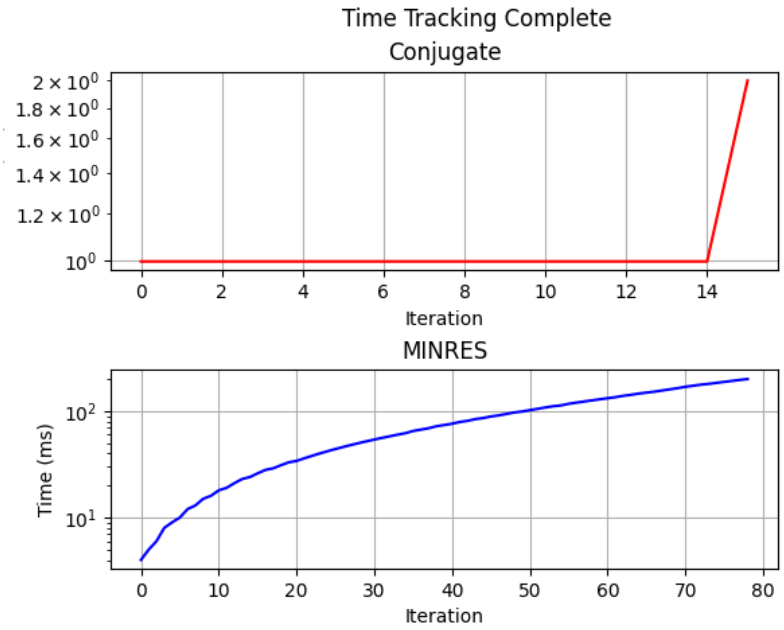
Each pair of charts shows the residual at every iteration of the specific topology of the graph with the Conjugate Gradient and the MINRES. The red line is a limit, and it is positioned on the x coordinate of the rank of the matrix of the linear problem because the convergence theory states that the methods usually arrive at the objective in less than the rank number. The algorithms are stopped in three cases:

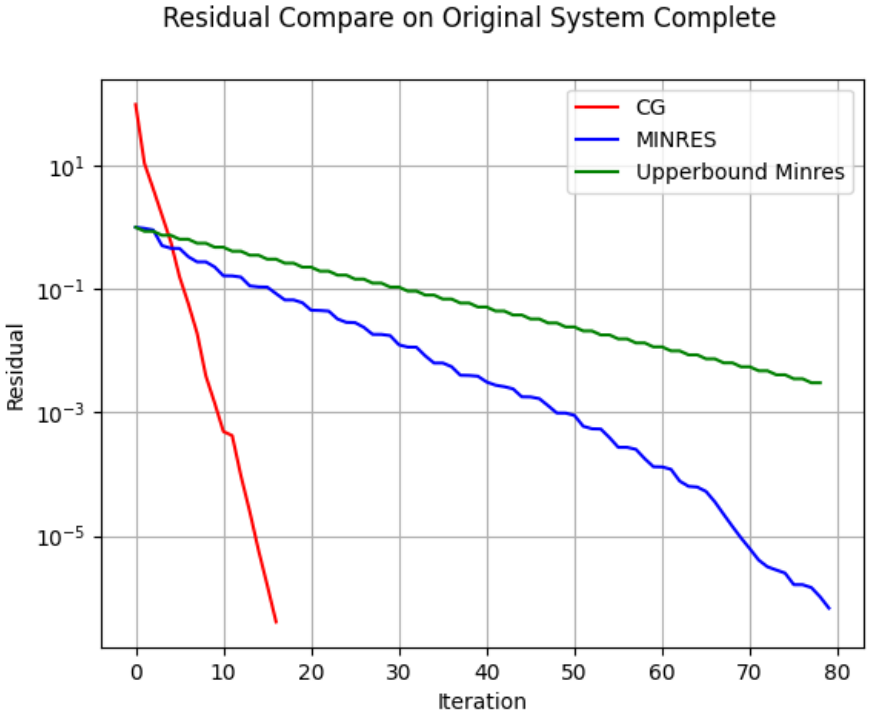
1. The number of the iteration are equal to the number of the matrix rank
2. The residual is less or equal to the 10^{-6}
3. If there is a lucky breakdown down, but it can be only in the MINRES method

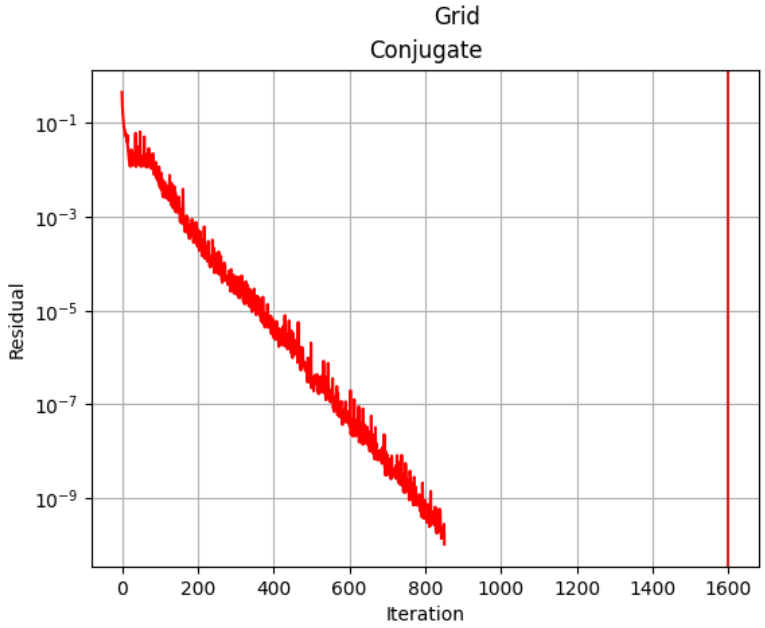
The instances of the problems include a different topology of the graph. The types we analyzed are the three we showed in the section 1.5. The first table shows the parameter generators used for every graph's topology. They set the dimensions of the matrices with the numbers of the arcs and the nodes.

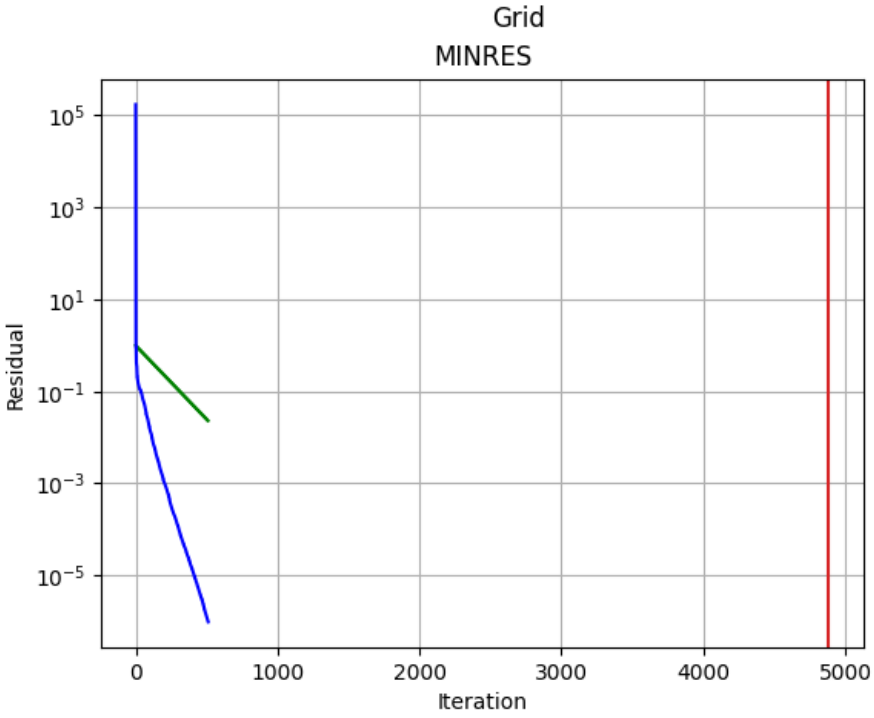

Complete
Conjugate
Residual
Iteration
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

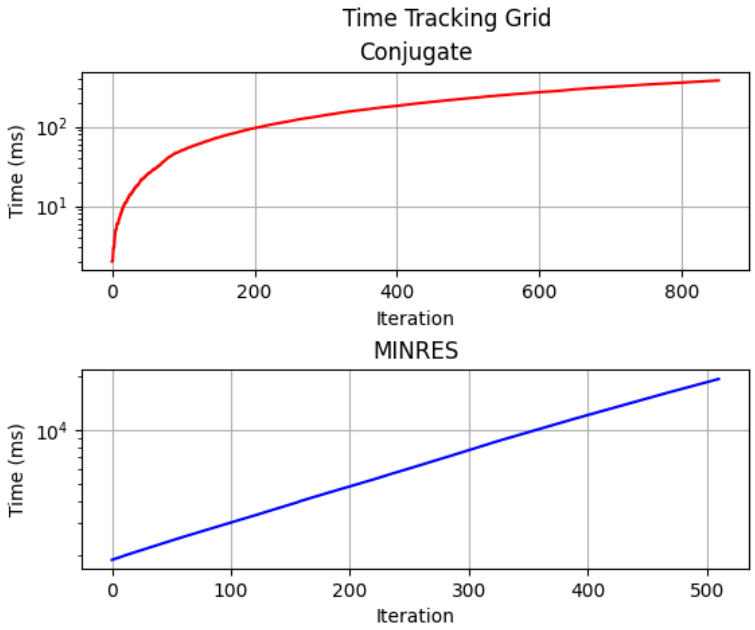

Complete
MINRES
Residual
Iteration
0
50
100
150
200

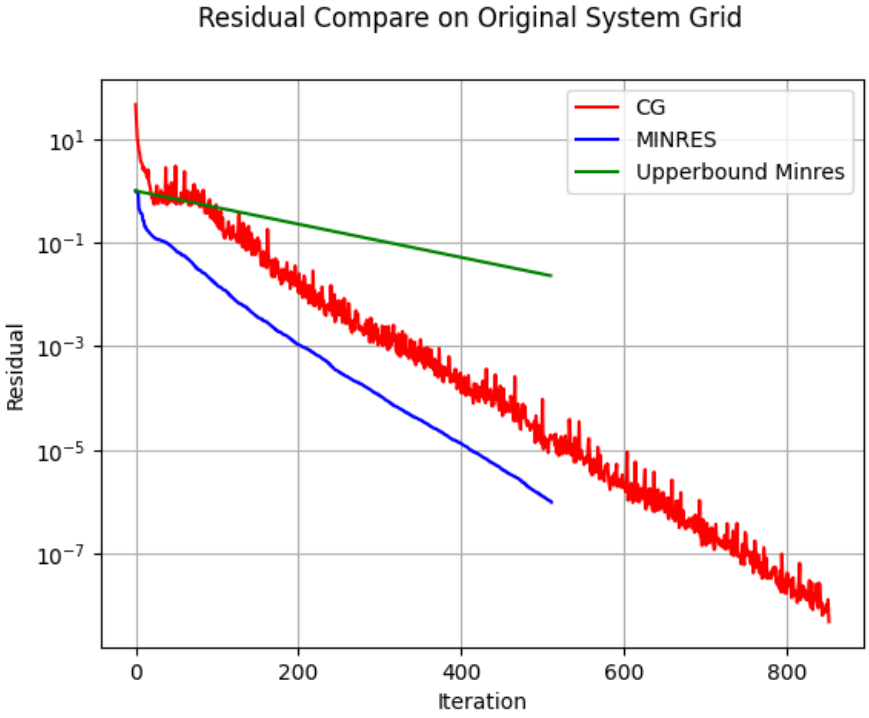

Time Tracking Complete
Conjugate
2×10^0
1.8×10^0
1.6×10^0
1.4×10^0
1.2×10^0
10^0
0
2
4
6
8
10
12
14
Iteration
MINRES
Time (ms)
10^2
10^1
0
10
20
30
40
50
60
70
80
Iteration

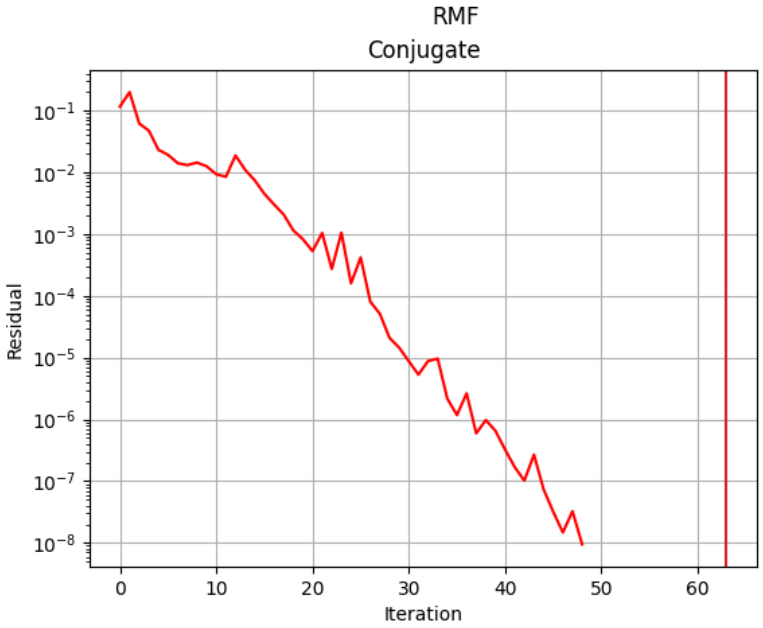

Residual Compare on Original System Complete
CG
MINRES
Upperbound Minres
Residual
10^1
10^{-1}
10^{-3}
10^{-5}
0
10
20
30
40
50
60
70
80
Iteration

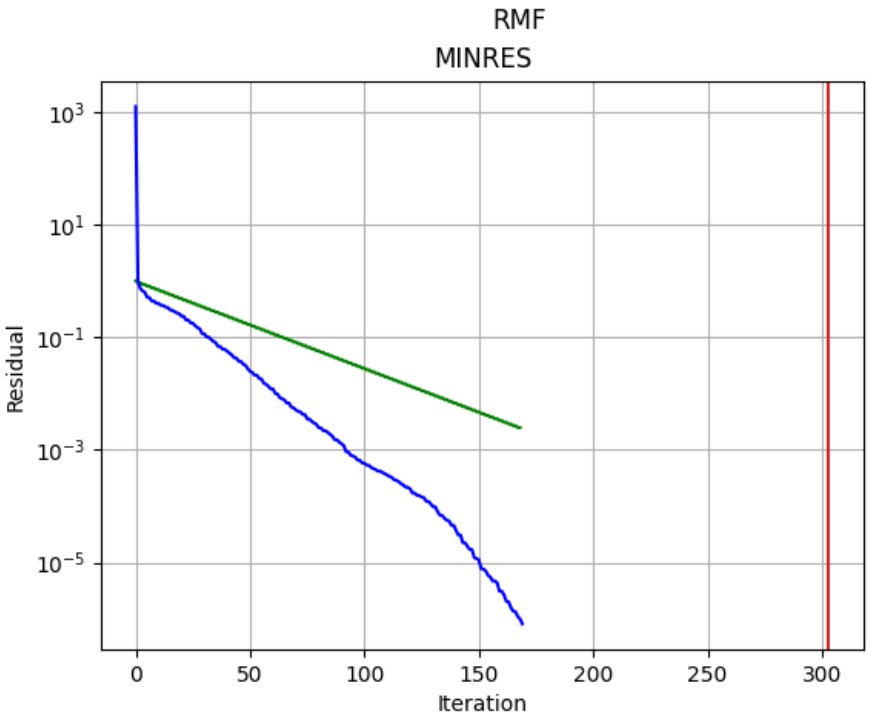

Grid
Conjugate
Residual
10^{-1}
10^{-3}
10^{-5}
10^{-7}
10^{-9}
0
200
400
600
800
1000
1200
1400
1600
Iteration

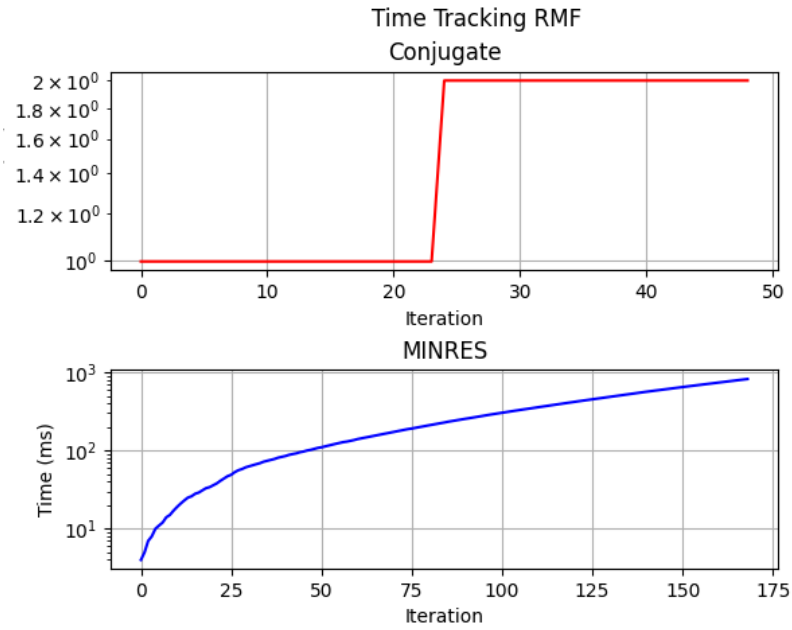

Grid
MINRES
Residual
10^{5}
10^{3}
10^{1}
10^{-1}
10^{-3}
10^{-5}
0
1000
2000
3000
4000
5000
Iteration

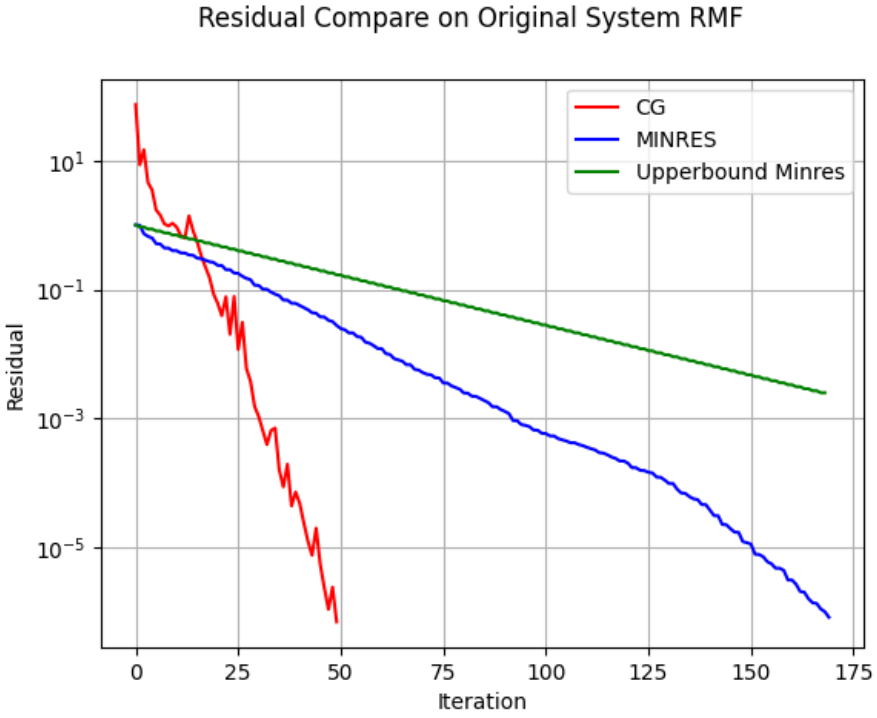

Time Tracking Grid
Conjugate
Time (ms)
10^2
10^1
0
200
400
600
800
Iteration
MINRES
Time (ms)
10^4
0
100
200
300
400
500
Iteration


Residual Compare on Original System Grid
CG
MINRES
Upperbound Minres
Residual
10^1
10^{-1}
10^{-3}
10^{-5}
10^{-7}
0
200
400
600
800
Iteration


RMF
Conjugate
Residual
Iteration
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
0
10
20
30
40
50
60


RMF
MINRES
Residual
Iteration
10^{3}
10^{1}
10^{-1}
10^{-3}
10^{-5}
0
50
100
150
200
250
300


Time Tracking RMF
Conjugate
2×10^0
1.8×10^0
1.6×10^0
1.4×10^0
1.2×10^0
10^0
0
10
20
30
40
50
Iteration
MINRES
10^3
10^2
10^1
Time (ms)
0
25
50
75
100
125
150
175
Iteration


Residual Compare on Original System RMF
CG
MINRES
Upperbound Minres
10^1
10^{-1}
10^{-3}
10^{-5}
Residual
0
25
50
75
100
125
150
175
Iteration

Configuration incidence matrices						
Graph	Nodes	Arcs	Cost Max/min	Capacity Max/min	Average Deficit	Total Deficit
Complete	20	190	100/50	100/50	5	20
Grid	1602	3284	10000/0	10000/0	/	113200
RMF	64	240	100/0	4/4	/	578

Results Original System					
Algorithm	Graph	Rank	Iterations	Time execution	Residual
CG	Complete	19	15	1.9996 ms	1.756888452337951e-07
MINRES	Complete	209	78	200.0005 ms	6.802886111049279e-07
CG	Grid	1601	851	380.002 ms	6.889403074114535e-07
MINRES	Grid	4885	510	19234.1601 ms	9.778825291572051e-07
CG	RMF	63	48	2.0002 ms	7.525496632728532e-07
MINRES	RMF	303	168	820.0239 ms	8.223245575896946e-07

The two methods have similar residual results, but the Conjugate Gradient is more efficient for execution time and memory than the MINRES because the first method can study the same matrices with the reduced system since the dimensions of the matrices are smaller than the original. However, the MINRES has better results in the study of the Grid Graph, and it is easier to apply than the hard condition of the Conjugate Gradient. The results respect our predictions because the number of iterations is lesser than the rank of the problem matrix.

References

[1] J. Lee, *A First Course in Linear Optimization*. Reex Press, 2013–2021. [Online]. Available: https://github.com/jon77lee/JLee_LinearOptimizationBook

[2] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Natl. Bur. Stand. (1934)*, vol. 49, no. 6, p. 409, Dec. 1952.

[3] A. S. Reddyh, "Number theory and graph theory." [Online]. Available: http://epgp.inflibnet.ac.in/epgpdata/uploads/epgp_content/S000025MS/P001478/M015498/ET/1462425168E-textofChapter9Module2.pdf

[4] Wikipedia, "Rank (linear algebra)." [Online]. Available: https://en.wikipedia.org/wiki/Rank_(linear_algebra)#

[5] W. E. Arnoldi, "The principle of minimized iterations in the solution of the matrix eigenvalue problem," *Quarterly of Applied Mathematics*, vol. 9, pp. 17–29, 1951. [Online]. Available: https://api.semanticscholar.org/CorpusID:115852469

[6] C. C. Paige and M. A. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM Journal on Numerical Analysis*, vol. 12, no. 4, pp. 617–629, 1975. [Online]. Available: https://doi.org/10.1137/0712047