Assignment: HALE for an UAV

Modelling in Aerospace Engineering

Assignment I

31^{st} of March, 2025

Authors:

Andrés Velázquez Vela
Sergio Viejo Casado

This report aims to explain the methods used to solve the systems and to ensure a proper understanding of the results.

1 Solving the linear system: $A_1 x_1 = b_1$.

1.1 LU decomposition with Gaussian Elimination.

So in this section, a variant of LU decomposition is used, also utilizing Gaussian elimination. This approach defines L as a lower triangular matrix with ones on its main diagonal. The matrix A is factored into $A = LU$, where L is lower triangular and U is upper triangular. This method can be found the official MATLAB documentation at `matlab.com`.

I´m going to briefly explain this method.

1. **Input Validation:** Check if A is square.
2. **Initialization:** $L = I_n, \quad U = A$
3. **LU Decomposition via Gaussian Elimination:** For $k = 1$ to $n - 1$:
   a. Check for zero pivot: If $U_{k,k} = 0,$ error: "Zero division!!!!!!!!!", we are going to solve it in 1.2.
   b. Compute multipliers and update L:

$$L_{i,k} = \frac{U_{i,k}}{U_{k,k}}, \quad \text{for } i = k+1, \ldots, n$$

   c. Update U by eliminating elements below the pivot:

$$U_{i,j} = U_{i,j} - L_{i,k} \cdot U_{k,j}, \quad \text{for } i = k+1, \ldots, n, \quad j = k, \ldots, n$$

The following output occours becouse of the divission by the $U_{k,k}$ pivot when it is near zero. We do not reach the desired tolerance.

```
1    Step 2: Residual |b1 - A1*x1|...
2    Maximum absolute residual: 3.159410e-03
```

With the pivoting process we are going to ensure that at each step the largest available pivot is selected to reduce the aproximation effect of division by near zero.

1.2 The addition of Pivoting.

The function employs **partial pivoting**, meaning it swaps rows but not columns. The diferent steps that from the 1.1 are:

$$PA = LU$$

1. For each k column k, we find the row i that contains the largest absolute value in (k, k):

$$\text{maxIndex} = \max_{i \geq k} |U(i, k)|$$

2. If the element is not in position (k, k), swap row k with row maxIndex in U and P:

$$U([k, \text{maxIndex}], :) = U([\text{maxIndex}, k], :)$$

$$P([k, \text{maxIndex}], :) = P([\text{maxIndex}, k], :)$$

3. We also apply it to L:

$$L([k, \text{maxIndex}], 1 : k-1) = L([\text{maxIndex}, k], 1 : k-1)$$

As seen in this case we can reach the desired tolerance because we lowered the effect of the division by a defect in a pivot.

```
Step 2: Residual |b1 - A1*x1|...
Maximum absolute residual: 1.776357e-15
```

2 Solving the linear system: $A_2 x_2 = b_2$.

2.1 Iterative solution with Jacobi's and Gauss-Seidel's methods

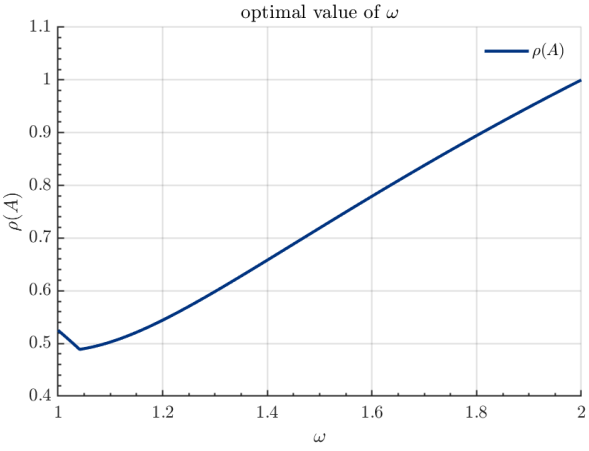
Both methods are stationary, which means they split A into $M + N$, and solve the system by iterating $x^{(k+1)} = x^{(k)} + M^{-1} r^{(k)}$. Convergence is linear in both cases. The condition for them to converge is A being strictly dominant or symmetric positive definite.

Nevertheless, Gauss-Seidel's typically converges faster (when it does) than Jacobi's because of using the newly computed values immediately.

The explanation behind Gauss seidel converging and jacobi not doing so is the spectral radius $p(T)$. It is defined by the minimum eigenvalue of the iteration matrix (T), which is different in both methods. Therefore one may converge $(p(T) < 1)$ while the other one does not $(p(T) > 1)$.

2.2 Successive Over Relaxation (SOR) method using the optimal ω value.

SOR is also a stationary method in which M and therefore T are modified by a factor w between 1 and 2 that is chosen to minimize $p(T)$. To do so we plotted $p(T)$ against w and chose the proper one ($w = 1.04104$). It is an acceleration of Gauss-Seidel's method as the lower the spectral radius, the faster it converges.



2.3 Gradient descent and conjugate gradient method, also preconditioning.

Gradient Descent method is based on converting the linear system into a minimization problem. It works by following the steepest descent direction in each iteration, which makes it a non-stationary method. It is expected to be slow for problems with many directions.

On the other hand, the Conjugate Descent method does not always descend in the steepest direction but rather chooses a search direction in each iteration re-minimizing in the previously searched directions. It adapts to the problem's curvature while Gradient Descent strategy is fixed, resulting in a faster convergence.

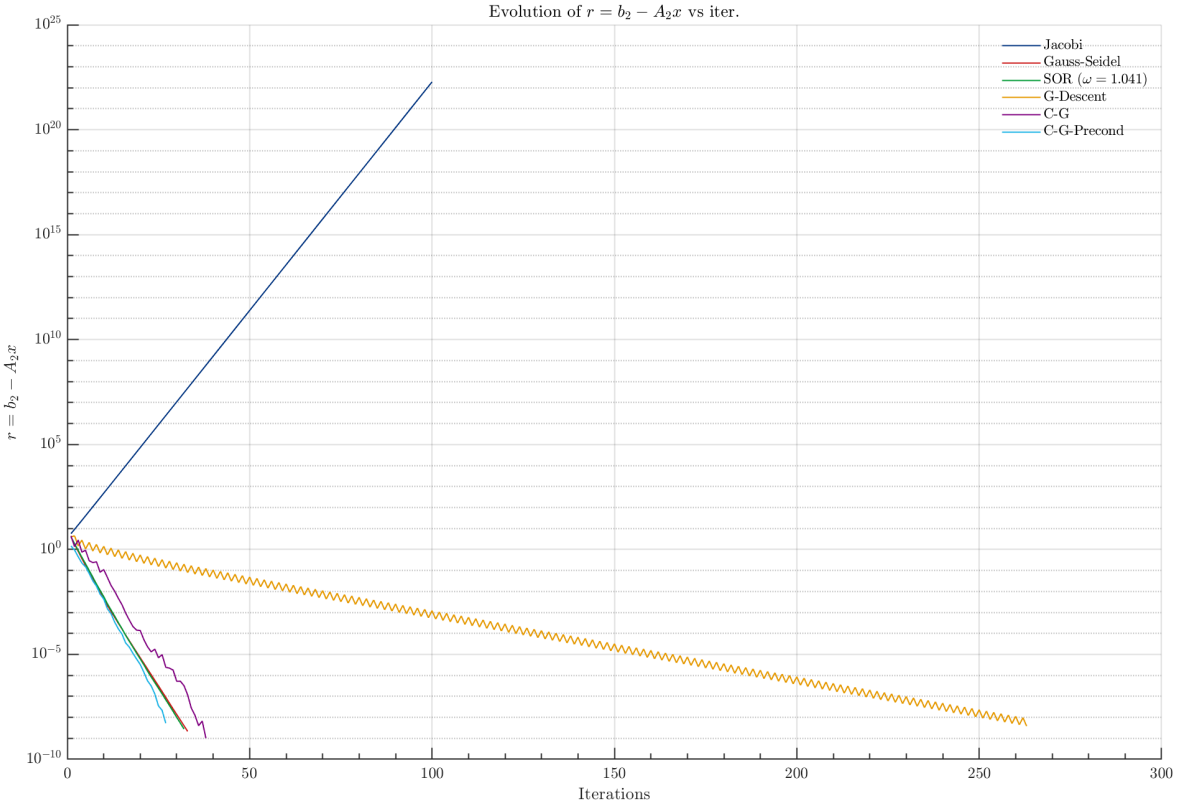
Lastly, preconditioning is used to reduce the condition number of the matrix, which is useful if $\text{Cond}(A) \gg 1$ as the number of iterations is proportional to its square root. It is essential in large problems.

2.4 Evolution of the norm of the residual $r = b_2 - A_2x$ vs number of iterations.

First looking into Jacobi and Gauss-Seidel and SOR methods, we observe linearity in all of them. As mentioned earlier, Jacobi does not converge, which may be caused by the spectral radius of the iteration matrix being greater than 1.

Regarding SOR with optimal w, it shows a slightly $(1,04 \sim 1)$ steeper slope than Gauss-Seidel's one, because of using w optimal. The Gradient descent presents a zig-zagging evolution due to the poorly

aligned search direction with respect to the problem's curvature. While the Conjugate Gradient Descent does not show them because it adapts to the problem's curvature.



The plot also shows how the preconditioning fastens the Conjugate Gradient Descent convergence as desired.

2.5 Number of iterations needed

	Jacobi	**Gauss-Seidel**	**SOR**	**Steepest Descent**	**CG**	**precondCG**
Number of Iterations	∞	33	32	263	38	27

Table 1: Comparison of iteration counts for different methods

With respect to the number of iterations, Jacobi reached the maximum without converging, likely due to the matrix nature, probably lacking diagonal dominance. Regarding Gauss Seidel and SOR, we find the optimal w barely improves the convergence, the reason behind is that the optimal w ($w = 1.04$) is already close to $w = 1$, which would mean gauss seidel is already close to using the optimal w and therefore, its best form.

Lastly, the gradient descent method needed the most iterations to converge, which was expected due to the number of dimensions. Accordingly, the conjugate gradient method achieved convergence faster thanks to its adaptability, and even faster when preconditioned, around 30% less iterations.