# GMRES Algorithm Writeup

## 1. GMRES Algorithm Overview

GMRES (Generalized Minimal Residual) is an iterative method for solving large, sparse, and non-symmetric linear systems of the form $Ax = b$. It belongs to the family of Krylov subspace methods. The core idea is to construct an approximate solution within a Krylov subspace such that the corresponding residual is minimized in the sense of the 2-norm.

- **Objective**: Given a normal matrix $A$ and a vector $u$, construct an **orthonormal basis** $Q$ for the Krylov subspace and the corresponding upper **Hessenberg matrix** $H$ using the **Arnoldi** iteration.
- **Step Summary**:

  1. **Initialization**: Normalize the vector $u$ to obtain $q_1$, and set it as the first column of the orthonormal basis matrix $Q$.

  2. **Iteration Process**: For $k = 1, 2, \ldots, m$, compute $v = Aq_k$. Then orthogonalize v using all previously obtained basis vectors $q_1, \ldots, q_k$:

     - For each $j = 1, \ldots, k$, compute $hj, k = q_j^T v$*, and then update* $v \leftarrow v - hj, kq_j$.

  3. **Normalization**: Set $hk + 1, k = |v|2.$ *If* $hk + 1, k \neq 0$*, define* $qk + 1 = v/h_{k+1,k}$, and add it to $Q$.

  4. **Termination Condition**: The iteration stops when either the relative residual norm falls below a specified tolerance or when the maximum number of iterations is reached.

## 2. Answer Details

### 2.1 Basics: Arnoldi iteration

### 2.1.1 Algorithm Description

Below is a Arnoldi iteration pseudocode.

```
1. Set $q_1 = u / \|u\|_2$ % Normalize initial vector to get first basis vector q1
2. for k = 1, 2, ..., m:
    a. Compute $v \leftarrow A \cdot q_k$  % Compute new vector
    b. Orthogonalize v against existing basis vectors:
      for j = 1, ..., k:
          $h_{jk} \leftarrow q_j^T \cdot v$
          $v \leftarrow v - h_{jk} \cdot q_j$
    c. Compute $h_{(k+1),k} \leftarrow \|v\|_2$ % Compute norm of the new vector and normalize
    d. if $h_{(k+1),k} = 0$ (happy breakdown occurs), stop iteration % Check
    e. Set $q_{k+1} \leftarrow v / h_{(k+1),k}$  % Normalize v to get the new basis vector
3. Return matrices Q (columns $q_1, q_2, \ldots$) and H (elements $h_{jk}$)
```

To improve numerical stability, I choose using the **Modified** Gram-Schmidt algorithm in this case. This step-2b immediately updates $v$ helping reduce round-off errors and improves numerical stability compared to the classical version.

### 2.1.2 Implementation Details

For this assignment, Key features of the implementation in MATLAB include:

**1.Modified Gram-Schmidt Process:**

The modified Gram-Schmidt procedure is used rather than the classical version to improve numerical stability. At each step, the algorithm immediately subtracts the projection onto each previously computed vector. This helps in reducing the loss of orthogonality that might occur in floating-point arithmetic.

**2.Detection of Breakdown:**

A tolerance level (set to $10^{-12}$) is used to determine whether the norm of the orthogonalized vector is too small. If this happens, it means the new vector is almost linearly dependent on previous vectors, a phenomenon known as "**happy breakdown**". Current Krylov subspace is already rich enough to capture the necessary information about A

The tolerance is chosen based on the **machine precision**. Although double precision arithmetic typically has an epsilon around $10^{-16}$, a higher tolerance is used to account for accumulated **round-off errors**.

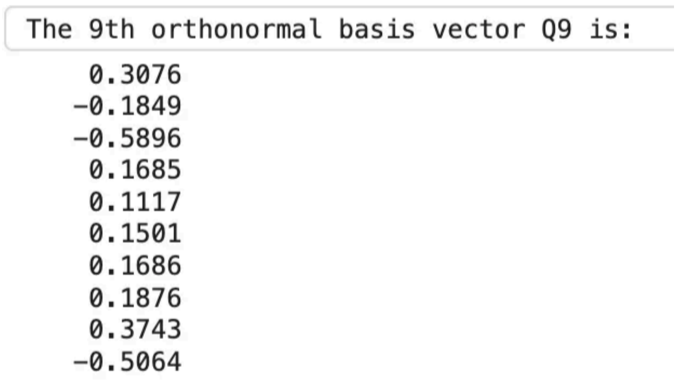
**3.Outputs of the Function:**

The function arnoldi returns:

- **OrthogonalBasis:** An $n \times (m+1)$ matrix containing the computed orthonormal basis vectors.
- **HessenbergMatrix:** A $(m+1) \times m$ upper Hessenberg matrix containing the projection coefficients.

### 2.1.3 Testing and Conclusion

I tested the Arnoldi iteration algorithm (using modified Gram-Schmidt) to check if it works correctly and remains stable. For the test, I used a 10×10 matrix within a randomly chosen initial vector. Within 9 iterations, it successfully generated the orthonormal basis and the Hessenberg matrix. The program output for the 9th orthonormal basis vector $Q_9$ is shown below:



```
The 9th orthonormal basis vector Q9 is:
    0.3076
   -0.1849
   -0.5896
    0.1685
    0.1117
    0.1501
    0.1686
    0.1876
    0.3743
   -0.5064
```

A quick calculation shows that the sum of squares of these elements is close to 1. This matches what we'd expect for an orthonormal vector, confirming that the algorithm is numerically stable. Also, no "happy breakdown" occurred during the test, meaning the iteration successfully completed all 9 steps and produced the 9th basis vector as planned.

## 2.2 Serial implementation of GMRES

### 2.2.1 Algorithm Description

Blow is a Serial GMRES pseudocode.

```
1. Set x0 = 0.                    % Initial guess is zero.
2. Compute r0 = b - A*x0 and beta = norm(r0). % Calculate initial residual & norn
3. Normalize: v1 = r0 / beta.        % Create the first basis vector.
4. Compute [Q, H] = arnoldi(A, r0, m). % Build the Krylov subspace using Arnoldi
5. For k = 1 to m do:
      a. Let H_k = first (k+1)×k block of H. % Extract current Hessenberg matrix.
      b. Let Q_k = first k columns of Q.  % Get current orthonormal basis.
      c. Set e1 = [beta; zeros(k,1)]. % Form right-hand side vector .
      d. Solve for y in H_k * y ≈ e1.  % Solve least squares problem.
      e. Update x_approx = x0 + Q_k * y.  % Compute approximate solution.
      f. Record resHist(k) = norm(b - A*x_approx). % Save current residual norm.
6. End For.
7. Return x_approx and resHist.  % Return final solution & residual history.
```

The GMRES algorithm aims to solve a linear system Ax = b by gradually reducing the residual error at each iteration.

### 2.2.2 Implementation Details

**1.Initialization:**

The function sets the initial guess $x_0$to zero, computes the initial residual $r_0 = b - Ax_0$, and uses its $norm \beta$ to generate the first normalized basis vector.

**2.Arnoldi and Least-Squares Update:**

An external Arnoldi function is called to build an orthonormal basis $Q$ and Hessenberg matrix $H$ for the Krylov subspace. For each iteration, the function extracts the current $Q_k$ and $H_k$ and solves the least squares problem $\min_y ||\beta e_1 - H_k y||$ to update the approximate solution.

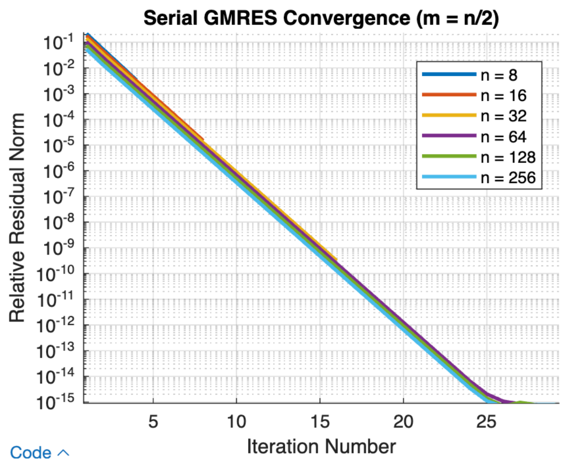
**3.Residual Tracking:**

After each iteration, the 2-norm of the residual $b - Ax_{\text{approx}}$ is computed and stored, providing a history of convergence to analyze the performance of GMRES.

### 2.2.3 Testing and Conclusion

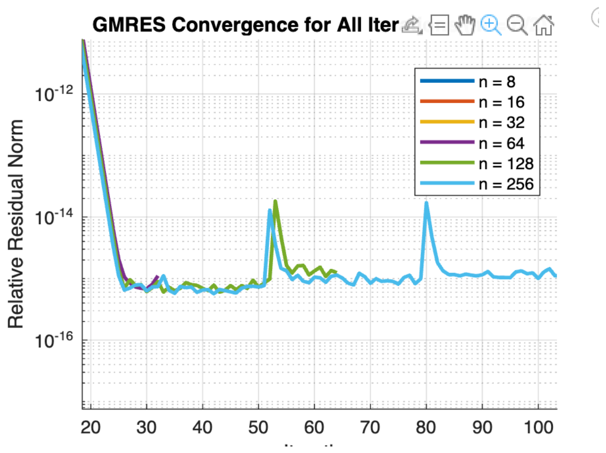
I tested the GMRES implementation on a tridiagonal system (main diagonal -4, off-diagonals 1) for $n = 8, 16, 32, 64, 128, 256$, each time allowing up to $m = n/2$ iterations. The three figures show the relative residual $\|r_k\|/\|b\|$ on a semilog scale across different iteration ranges:

#### 1.Early Iterations



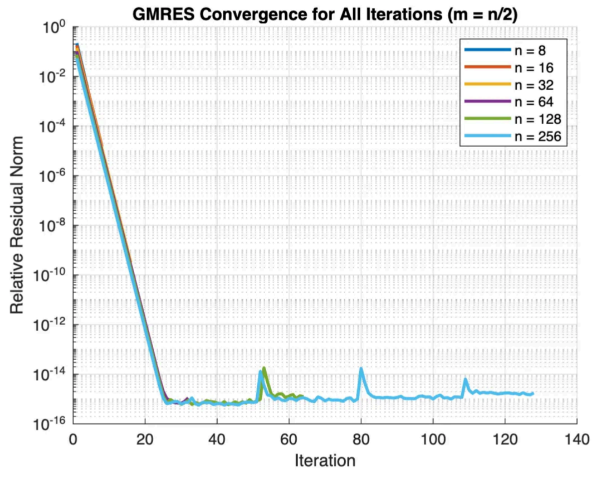
A. In the first 10–20 steps, all curves drop very quickly from about $10^0$ down to around $10^{-14}$or lower.
B. This indicates that GMRES handles this tridiagonal system well, rapidly decreasing the error in just a few iterations.

#### 2.Later Iterations



A. Once the residual is near machine precision ($\approx 10^{-16}$), small spikes appear in some of the curves (especially for larger n like 128 and256).
B. These small jumps are likely due to floating-point effects or near-rank-deficient least-squares solves. They do not mean the method fails because the residual still remains extremely small.

### 3.Overall Convergence



A. GMRES converges very fast for all n, showing that this problem is not particularly difficult.
B. The occasional spikes do not affect the final accuracy since the residual stays close to machine precision.
C. In a practical setting, one would typically stops well before reaching such a tiny residual.

## 2.3 Parallel implementation of GMRES.

### 2.3.1 Algorithm Description

Blow is a Parallel GMRES pseudocode.

```
1. Set x0 = 0.
2. Compute r0 = b - A*x0 and beta = norm(r0).
3. Normalize: v1 = r0 / beta.
% Build Krylov subspace using a parallel Arnoldi process.
4. Compute [Q, H] = arnoldiParallel(A, r0, m).
5. For k = 1 to m do:
    a. Let H_k = first (k+1)×k block of H. % Extract current Hessenberg matrix.
    b. Let Q_k = first k columns of Q.     % Get current orthonormal basis.
    c. Set e1 = [beta; zeros(k,1)].     % Form right-hand side vector.
    d. Solve for y in H_k * y ≈ e1.      % Solve least squares problem.
    e. Update x_approx = x0 + Q_k * y.    % Compute approximate solution.
    f. Record resHist(k) = norm(b - A*x_approx). % Save current residual norm.
    g. If resHist(k)/norm(b) < tol, break.   % Stop if it’s below tolerance.
```

```
6. End For.
7. Return x_approx and resHist.
```

In our parallel version of GMRES, we focus on speeding up the most time-consuming steps of the Arnoldi iteration.

## 2.3.2 Implementation Details

### 1. Inner Product Computations

For the Gram–Schmidt step, where we compute $h_{j,k} = q_j^\top w$, we perform these dot products in parallel(eg. using MATLAB's parfor or MPI approach or OpenMP in C).

In this case, using MATLAB's parfor loop for inner products allows each worker to calculate part of the dot product independently, and then a parallel reduction (or summing of the partial results) is used to obtain the final value. This parallelization is critical because these operations can be very time-consuming for large matrices.

### 2. Robust Stopping Criteria

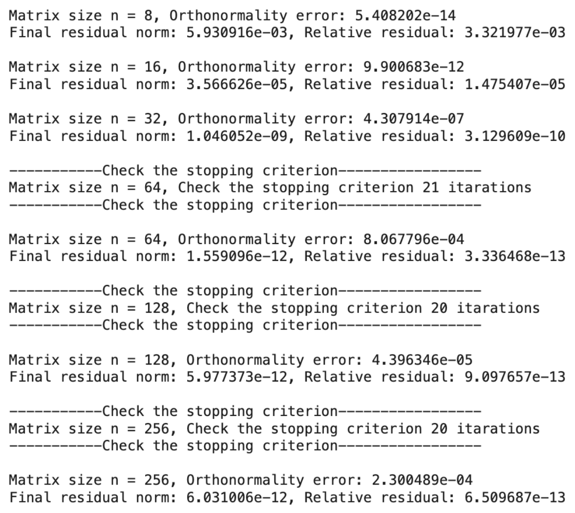
A good stopping criterion for a parallel GMRES algorithm is to monitor the R**elative Residual Norm**. We stop when $\|r_k\|/\|b\| < \mathrm{tol}$, for instance $\mathrm{tol} = 10^{-12}$. This is often the most straightforward measure of accuracy. It ensures that once the solution is accurate enough relative to the size of b, the algorithm stops, avoiding unnecessary iterations.

### 3. Correctness Tests

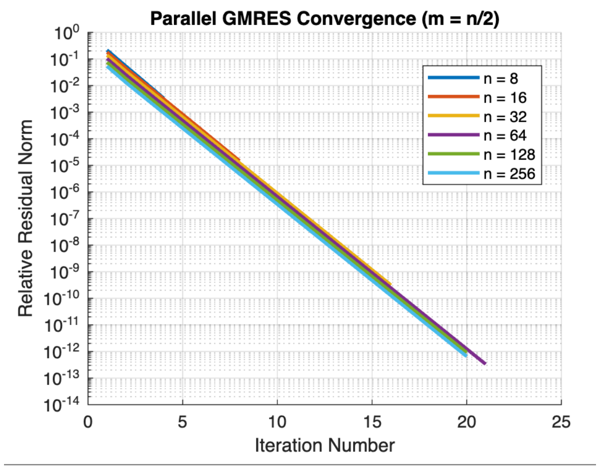
- **Orthonormality Check**: After constructing the Krylov subspace using Arnoldi, we compute $\|Q^\top Q - I\|$ to verify that the basis vectors Q remain close to orthonormal. A small value (like $10^{-12} to 10^{-4}$) indicates that numerical errors have not significantly damaged orthogonality.
- **Residual Verification:** At the end of the iterations, we print the final absolute residual $\|b - Ax_{\text{approx}}\|$ and the relative residual $\|b - Ax_{\text{approx}}\|/\|b\|$. If these norms drop below a chosen tolerance (like $10^{-12}$), it confirms that GMRES has converged to an correct solution.

## 2.3.3 Testing and Conclusion

From  parallel GMRES **Correctness output** for different matrix sizes n here：



```
Matrix size n = 8, Orthonormality error: 5.408202e-14
Final residual norm: 5.930916e-03, Relative residual: 3.321977e-03

Matrix size n = 16, Orthonormality error: 9.900683e-12
Final residual norm: 3.566626e-05, Relative residual: 1.475407e-05

Matrix size n = 32, Orthonormality error: 4.307914e-07
Final residual norm: 1.046052e-09, Relative residual: 3.129609e-10

-----------Check the stopping criterion-----------------
Matrix size n = 64, Check the stopping criterion 21 itarations
-----------Check the stopping criterion-----------------

Matrix size n = 64, Orthonormality error: 8.067796e-04
Final residual norm: 1.559096e-12, Relative residual: 3.336468e-13

-----------Check the stopping criterion-----------------
Matrix size n = 128, Check the stopping criterion 20 itarations
-----------Check the stopping criterion-----------------

Matrix size n = 128, Orthonormality error: 4.396346e-05
Final residual norm: 5.977373e-12, Relative residual: 9.097657e-13

-----------Check the stopping criterion-----------------
Matrix size n = 256, Check the stopping criterion 20 itarations
-----------Check the stopping criterion-----------------

Matrix size n = 256, Orthonormality error: 2.300489e-04
Final residual norm: 6.031006e-12, Relative residual: 6.509687e-13
```



**1.Orthonormality Error**

•For n = 8 and n = 16, the error is extremely small (on the order of $10^{-14}$ to $10^{-12}$), showing that the basis vectors remain highly orthonormal for small systems.

•As n increases, the error generally grows (e.g., it can reach $10^{-4}$ for n = 256), which is common in larger subspaces because rounding errors accumulate.

**2.Final Residual Norm**

•For n = 8, the final residual is about $5.93 \times 10^{-3}$, which is not very small.

•For n = 16 and higher, the residual quickly drops to $10^{-5}, 10^{-9}, 10^{-12}$, or even lower, showing that GMRES can achieve higher accuracy for larger n. So for larger n, the Krylov subspace is bigger and can approximate the solution more effectively.

**3. Relative Residual**

•The relative residual varies from about $10^{-3}$ to $10^{-5}$ or even $10^{-13}$, reflecting different final accuracies for different n.

•When the relative residual reaches $10^{-10}$ or $10^{-13}$, it is close to machine precision, meaning the numerical solution is extremely accurate.

## Overall Observations

The serial and parallel versions **produce similar convergence curves,** which is expected as they solve the same mathematical problem. The parallel version leverages parfor for inner product computations to speed up the process on larger matrices.