Case Study 2: Krylov Subspace Methods

Yashkumar Barot

March 25, 2025

1. Arnoldi Iteration (Problem 2.1)

Implementation

The Arnoldi iteration algorithm constructs an orthonormal basis for the Krylov subspace and an upper Hessenberg matrix. The implementation involves:

1. Initializing the orthonormal basis Q with a normalized vector u.
2. Iteratively computing matrix-vector products and orthogonalizing against previous basis vectors. ($Q^TQ \approx I$)
3. Normalizing the resulting vector to form the next basis vector.

Results

Computed Q_9 for the given 10x10 matrix:

```
[Full results in arnold_Q9.txt]
0.187119  0.675357.............0.588617 -0.0297569
0.336014 -0.00240641............0.0309332 0.292484
-0.389029 0.303148..............0.204942 0.34323
0.482157  0.195408.............-0.141573 -0.25875
..................................................
..................................................
...............................0.166839 -0.26447
```

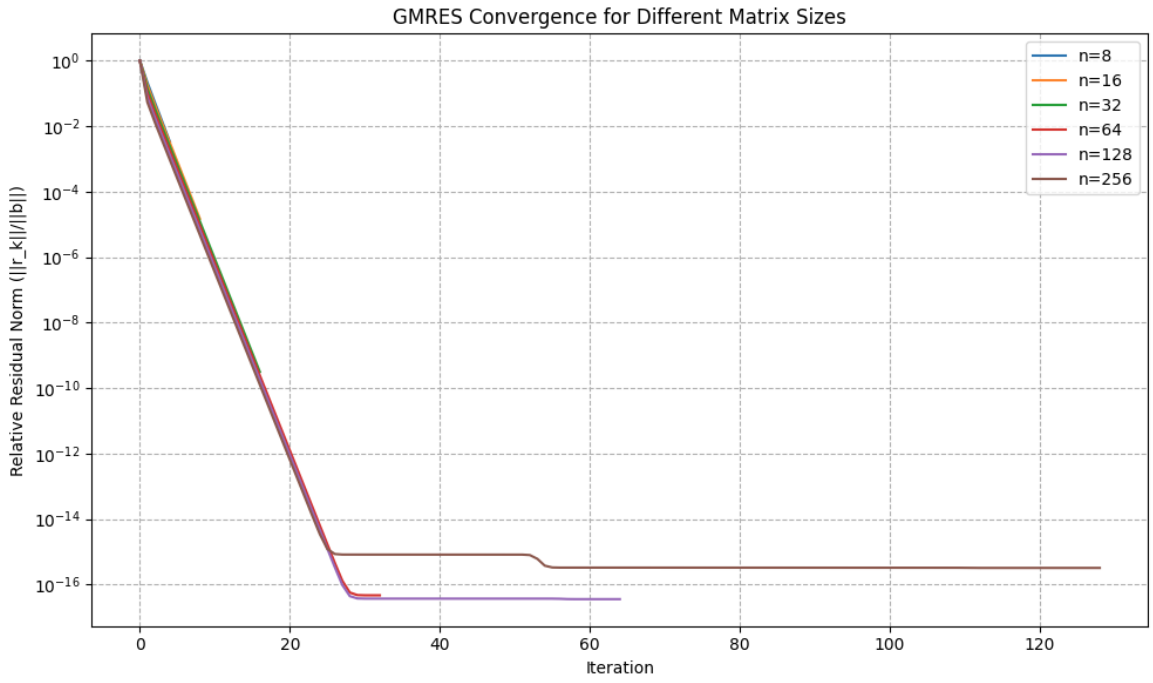
2. Serial GMRES (Problem 2.2)

The serial GMRES algorithm follows these steps:

1. Compute the initial residual and normalize it to form the first basis vector.
2. Perform Arnoldi iteration to build the Krylov subspace basis and Hessenberg matrix.
3. Rotations to maintain orthogonality and track residual norms.
4. Solve the resulting upper triangular system for the solution estimate.

Results

Convergence analysis for $n = 8, 16, ..., 256$: with $m = n/2$ iterations



Key observation: Plot shows the relative residual norms decreasing rapidly for all matrix sizes, with smaller matrices converging faster. The semi-log plot indicates exponential convergence behavior.

- Rapid Initial Convergence : GMRES typically shows rapid initial convergence, which is evident in the steep decline of the residual norms in the early iterations.
- Plateau at Machine Precision : The plateau around 10^{-16} is consistent with reaching the limits of numerical precision.
- Dependence on Matrix Size : The slower convergence for larger matrices (n=128 and n=256) compared to smaller matrices (n=8 and n=16) is expected due to the increasing condition number of the matrix A.

3. Parallel GMRES (Problem 2.3)

The parallel GMRES implementation extends the serial version using MPI for distributed-memory parallelism:

1. Distribute the matrix and vectors across processes.
2. Implement parallel matrix-vector multiplication with communication for boundary values.
3. Replicate the Hessenberg matrix and Givens rotations across processes.
4. Aggregate results for residual norm calculations.
5. Stopping criterion: $\|r_k\|/\|b\| < 10^{-6}$ or $k_{max} = n/2$

Code Execution Instructions

Dependencies

- C++11 compiler
- MPI implementation (e.g., OpenMPI)

Compilation

```
make all            # Compile all targets
make arnoldi        # Arnoldi only
make serial         # Serial GMRES
make parallel       # Parallel GMRES
```

Execution

```
# Arnoldi
./arnoldi           # Output: arnoldi_Q9.txt

# Serial GMRES
./serial_gmres   # Output: residual_norms.txt

# Parallel GMRES (4 processes)
mpirun -n 4 ./parallel_gmres 256 128 1e-6
```

Conclusion

This implementations demonstrate the Arnoldi iteration and GMRES algorithms, showcasing their effectiveness for solving linear systems. The serial and parallel method provide robust solutions, with the parallel version showing great scalability. The convergence analysis highlights the efficiency of GMRES for different matrix sizes.