universität wien

Bachelorarbeit

Implementation and experimental comparison between the COMMUNICATION AVOIDING-GENERALIZED MINIMAL RESIDUAL METHOD and standard GMRES

Verfasser
Robert Ernstbrunner

angestrebter akademischer Grad
Bachelor of Science (BSc)

Wien, 2018

Studienkennzahl lt. Studienblatt:	A 033 521
Fachrichtung:	Informatik - Scientific Computing
Betreuerin / Betreuer:	Univ.-Prof. Dipl.-Ing. Dr. Wilfried Gansterer, M.Sc.

Contents

Abstract

1 Introduction

2 Notation

explain matrix / graph notation

3 Related work

4 Computational kernels

introduction; definitions: kernel, communication-avoiding

4.1 Matrix powers kernel (MPK)

[3] p.60
Power iteration, SpMV instead of MV, sparse matrix like a graph → spacial, temporal locality not as efficiently used as in dense MV. Avoid communication by sending / receiving all necessary values beforehand (look at reachability of graph(A)) and computing s basis vectors without further communication.

4.2 Tall and skinny QR (TSQR)

Unconditionally stable like Householder QR but less communication

4.3 Block Classical Gram-Schmidt (BCGS)

Dense matrix-matrix multiplications → less communication (factor $\Theta(s)$ fewer messages) than unblocked CGS. Also, BCGS requires less communication that BMGS. No reorthogonalization because solving linear system with (std. GMRES uses unblocked MGS-Arnoldi) CA-GMRES uses TSQR which improves orthogonality of the block columns.

5 Arnoldi iteration

introduction.. produces $AQ = QH$

5.1 Arnoldi(s)

5.1.1 The Monomial basis

like powermethod, rapidly growing condition number and vector length → scale vector to lengh 1, but: basis vectors are always linearly independent in exact arithmetic. In machine precision they become inevitably dependent at a certain point → need a different basis.

5.1.2 The Newton basis

polinomial interpolation at shifts θ_1, θ_2, ..., θ_s.

Choosing the shifts find estimates of the eigenvalues of A (Ritz values).

The modified Leja ordering

Avoiding complex arithmetic

5.2 Arnoldi(s,t)

5.2.1 Introduction

5.2.2 Scaling the first basis vector

Arnoldi(s,t) produces $\underline{\Omega}$ which differs from MGS-Arnoldi $\hat{\underline{\Omega}}$ by a unitary scaling $\underline{\Theta} = \text{diag}(\theta_1, \theta_2, \ldots, \theta_{st}, \theta_{st+1})$ such that $\hat{\underline{\Omega}} = \underline{\Omega\Theta}$. $\rightarrow$

- QR factorization must not change direction of the first column
- compute $\theta_1 = \langle r_0, q_1 \rangle \ / \ \beta$
- compute q_1 via MGS-Arnoldi (this happens naturally when the first outer iteration is started with MGS-Arnoldi in order to compute Ritz values for the Newton basis)

5.2.3 QR factorization update

overlapping / non overlapping approach.

5.2.4 Updating the upper Hessenberg matrix

flop optimization. how to apply Givens rotations

6 CA-GMRES

uses Arnoldi(s,t) show pseudo code of implemented algorithm

6.1 Preconditioning

Left, right, split, we consider left preconditioning ($M^{-1}Ax = M^{-1}b$) only. Scaling is a special type of preconditioning. [3] considered two types of scaling in order to prevent rapid basis vector growth:

1. Balancing: replacing A by $A' = DAD^{-1}$ with D diagonal.
2. Equilibration: replacing A by $A' = D_r A D_c$ with D_r and D_c diagonal.

Algorithm 1 Newton CA-GMRES

Require: $n \times n$ linear system $Ax = b$ and initial guess x_0

$r_0 := b - Ax_0$, $\beta := \|r_0\|_2$, $q_1 := r_0/\beta$
for $k = 0$ to $t - 1$ do **do**
  if $k = 0$ **then**
    Compute $\underline{Q}_0$ and $\underline{H}_0$ using standard Arnoldi
    Set $\underline{\mathfrak{Q}}_0 := \underline{Q}_0$ and $\underline{\mathfrak{H}}_0 := \underline{H}_0$
    Compute Ritz values from H_0 and fix basis conversion matrix $\underline{B}_k$
    Reduce $\underline{H}_0$ from upper Hessenberg to upper triangular form using s Givens rotations $G_1, G_2, \ldots, G_s$. Apply the same rotations in the same order to βe_1, resulting in the length $s + 1$ vector ζ_0.
  else
    Compute $\underline{\acute{V}}_k$ using SpMV and one AXPY
    $\underline{\acute{\mathfrak{R}}}_{k-1,k} := \underline{\mathfrak{Q}}_{k-1}^T \underline{\acute{V}}_k$
    $\underline{\acute{V}}_k' := \underline{\acute{V}}_k - \underline{\mathfrak{Q}}_{k-1} \underline{\acute{\mathfrak{R}}}_{k-1,k}$
    Compute QR factorization of $\underline{\acute{V}}_k' \to \underline{\acute{Q}}_k \underline{\acute{R}}_k$
    Compute $\underline{\mathfrak{H}}_{k-1,1} := -\mathfrak{H}_{k-1} \mathfrak{R}_{k-1,k} R_k^{-1} + \underline{\mathfrak{R}}_{k-1,k} \underline{B}_k R_k^{-1}$
    Compute $H_k := R_k B_k R_k^{-1} + \tilde{\rho}_k^{-1} b_k z_k e_s^T - h_{k-1} e_1 e_{s(k-1)}^T \mathfrak{R}_{k-1,k} R_k^{-1}$
    Compute $h_k := \tilde{\rho}_k^{-1} \rho_k b_k$
    $\underline{\mathfrak{H}}_k := \begin{pmatrix} \mathfrak{H}_{k-1} & \underline{\mathfrak{H}}_{k-1,k} \\ h_0 e_1 e_{sk}^T & H_k \\ 0_{1,sk} & h_k e_s^T \end{pmatrix}$
    Apply Givens rotations $G_1, \ldots, G_{sk}$ in order to $\begin{pmatrix} \underline{\mathfrak{H}}_{k-1,k} \\ \underline{H}_k \end{pmatrix}$.
    Reduce $\underline{H}_k$ to upper triangular form using s Givens rotations $G_{sk+1}, \ldots, G_{s(k+1)}$. Apply the rotations in the same order to $\begin{pmatrix} \zeta_{k-1} \\ 0_s \end{pmatrix}$, resulting in the length $s(k+1)+1$ vector ζ_k.
  end if
  Element $s(k+1)+1$ of ζ_k is the 2-norm (in exact arithmetic) of the current residual $r_{k+1} = b - Ax_{k+1}$ of the current solution x_{k+1}.
  if converged **then**
    Use the above reduction of $\underline{\mathfrak{H}}_k$ to upper triangular form and ζ_k to solve $y_k := \operatorname{argmin}_y \|\mathfrak{H}_k y - \beta e_1\|_2$
    Set $x_k := x_0 + \mathfrak{Q}_k y_k$, and exit
  end if
end for

In their experiments solving nonsymmetric linear systems with CA-GMRES [3] found that for practical problems, equilibration established to be quite effective and almost made the basis type irrelevant. We observed something similar after applying the ILU(0) preconditioner to the system.

6.1.1 ILU(0) preconditioner

$M = LU$
in CA-GMRES apply M^{-1} to $r_0 = b - Ax_0$ at initialization and to $q_i = Aq_{i-1}$ int the MPK.

6.1.2 CA-ILU(0) preconditioner

summarize [2] (can be very long or short, dependent on overall length)

6.2 Convergence metrics

CA-GMRES produces cheap convergence metric, namely the relative residual $\|r_{k+1}\|_2$ / $\|r_0\|_2$. Might not be the best choice, depends too much on initial guess x_0.

If

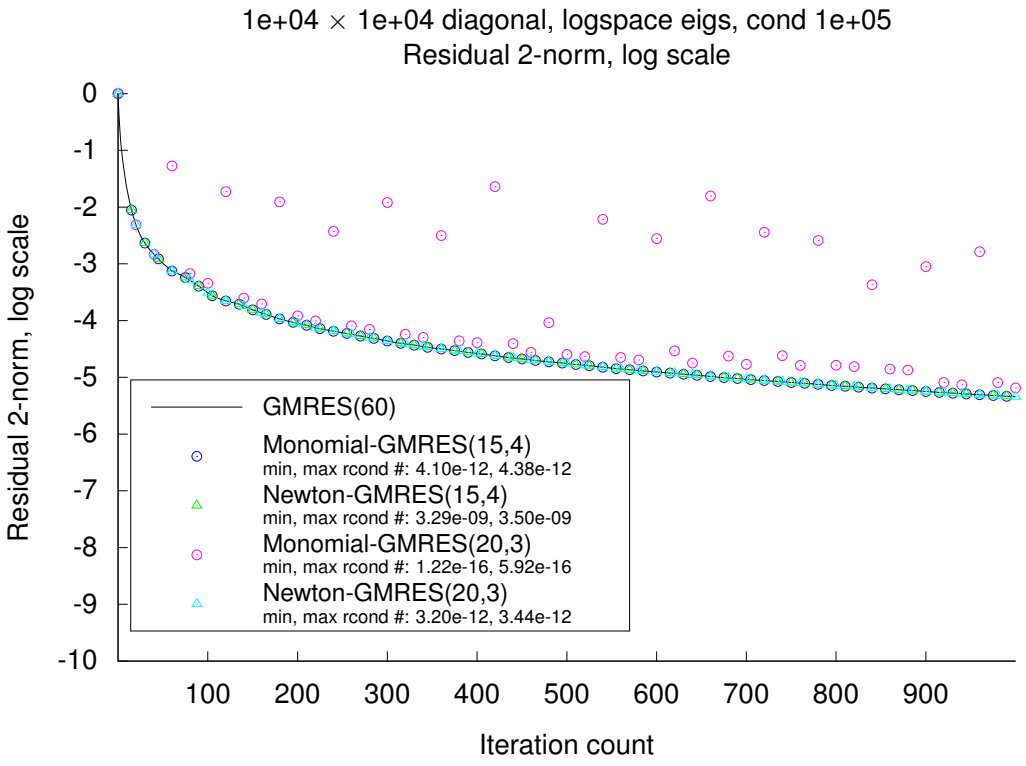
- $\|x_0\|_2$ too large $\rightarrow \|r_0\|$ will be large and iteration will stop too early.
- $x_0 = 0$ harder to make the relative residual small if A is ill-conditioned and x_{k+1} lies nearly in the nullspace of A.

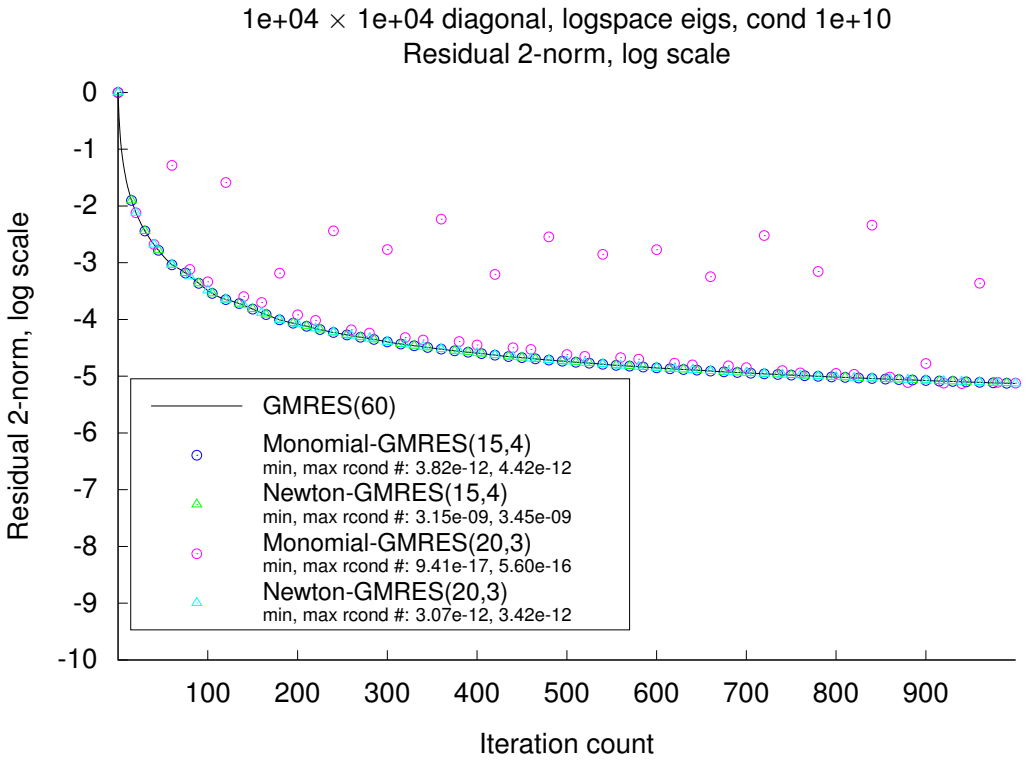
6.3 Implementation details

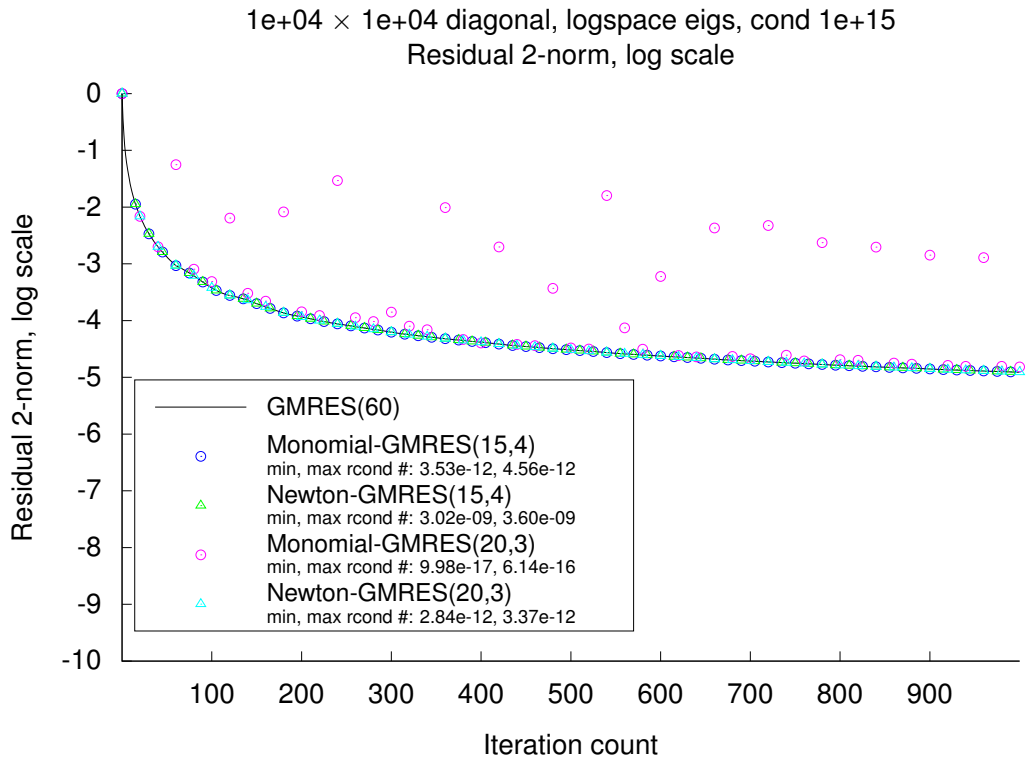
language: C++, libraries: intel MKL, profiler: ??? (intel VTune Amplifier, TAU, ...) could not implement neither the MPK nor the CA-ILU(0) preconditioner due to time constraints. Also Modified Leja ordering does not deal with under / overflow in the product to maximize like in [3].

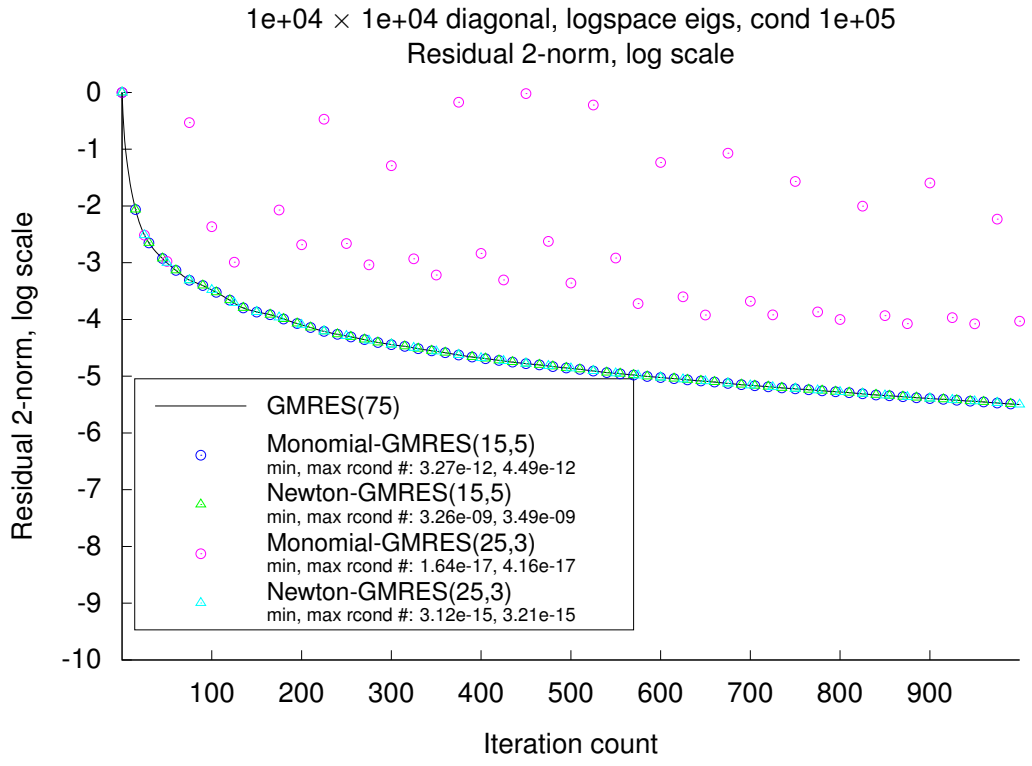
6.4 Numerical experiments

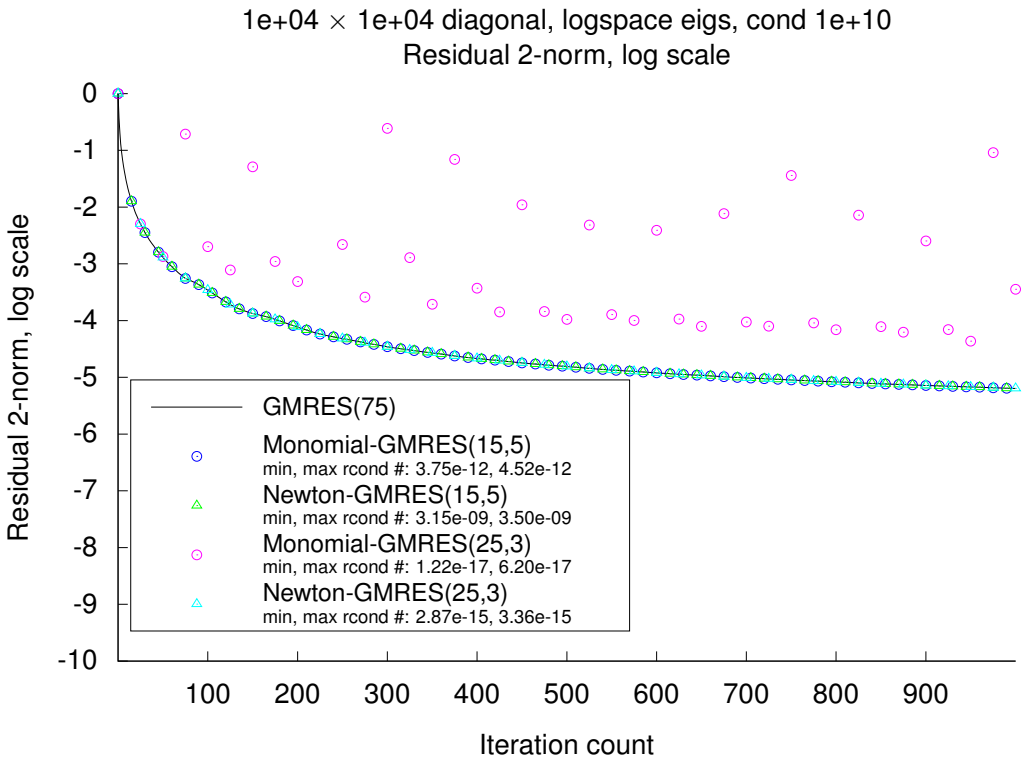
How the true solution $\hat{x}$ was generated: $\hat{x}(k) = u(k) + sin(2\pi k/n)$, where the scalar $u(k)$ is chosen from a random uniform [-1, 1] distribution. $\hat{x}$ was chosen in this way because a completely random solution is usually nonphysical, but a highly nonrandom solution (such as a vector of all ones) might be near an eigenvector of the matrix (which would result in artificially rapid convergence of the iterative method).

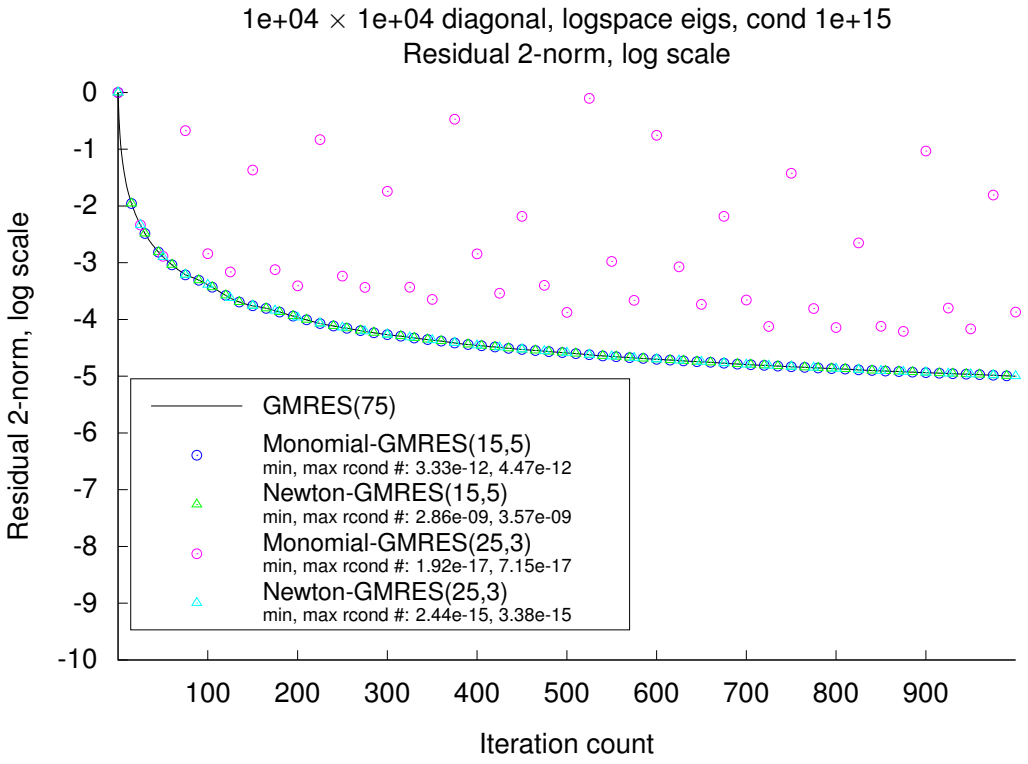

1e+04 × 1e+04 diagonal, logspace eigs, cond 1e+05
Residual 2-norm, log scale
Residual 2-norm, log scale
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
100
200
300
400
500
600
700
800
900
Iteration count
GMRES(60)
Monomial-GMRES(15,4)
min, max rcond #: 4.10e-12, 4.38e-12
Newton-GMRES(15,4)
min, max rcond #: 3.29e-09, 3.50e-09
Monomial-GMRES(20,3)
min, max rcond #: 1.22e-16, 5.92e-16
Newton-GMRES(20,3)
min, max rcond #: 3.20e-12, 3.44e-12


1e+04 × 1e+04 diagonal, logspace eigs, cond 1e+10
Residual 2-norm, log scale
Residual 2-norm, log scale
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
100
200
300
400
500
600
700
800
900
Iteration count
GMRES(60)
Monomial-GMRES(15,4)
min, max rcond #: 3.82e-12, 4.42e-12
Newton-GMRES(15,4)
min, max rcond #: 3.15e-09, 3.45e-09
Monomial-GMRES(20,3)
min, max rcond #: 9.41e-17, 5.60e-16
Newton-GMRES(20,3)
min, max rcond #: 3.07e-12, 3.42e-12


1e+04 × 1e+04 diagonal, logspace eigs, cond 1e+15
Residual 2-norm, log scale
Residual 2-norm, log scale
Iteration count
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
100
200
300
400
500
600
700
800
900
GMRES(60)
Monomial-GMRES(15,4)
min, max rcond #: 3.53e-12, 4.56e-12
Newton-GMRES(15,4)
min, max rcond #: 3.02e-09, 3.60e-09
Monomial-GMRES(20,3)
min, max rcond #: 9.98e-17, 6.14e-16
Newton-GMRES(20,3)
min, max rcond #: 2.84e-12, 3.37e-12


1e+04 × 1e+04 diagonal, logspace eigs, cond 1e+05
Residual 2-norm, log scale
Residual 2-norm, log scale
Iteration count
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
100
200
300
400
500
600
700
800
900
GMRES(75)
Monomial-GMRES(15,5)
min, max rcond #: 3.27e-12, 4.49e-12
Newton-GMRES(15,5)
min, max rcond #: 3.26e-09, 3.49e-09
Monomial-GMRES(25,3)
min, max rcond #: 1.64e-17, 4.16e-17
Newton-GMRES(25,3)
min, max rcond #: 3.12e-15, 3.21e-15


1e+04 × 1e+04 diagonal, logspace eigs, cond 1e+10
Residual 2-norm, log scale
Residual 2-norm, log scale
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
100
200
300
400
500
600
700
800
900
Iteration count
GMRES(75)
Monomial-GMRES(15,5)
min, max rcond #: 3.75e-12, 4.52e-12
Newton-GMRES(15,5)
min, max rcond #: 3.15e-09, 3.50e-09
Monomial-GMRES(25,3)
min, max rcond #: 1.22e-17, 6.20e-17
Newton-GMRES(25,3)
min, max rcond #: 2.87e-15, 3.36e-15


1e+04 × 1e+04 diagonal, logspace eigs, cond 1e+15
Residual 2-norm, log scale
Residual 2-norm, log scale
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
100
200
300
400
500
600
700
800
900
Iteration count
GMRES(75)
Monomial-GMRES(15,5)
min, max rcond #: 3.33e-12, 4.47e-12
Newton-GMRES(15,5)
min, max rcond #: 2.86e-09, 3.57e-09
Monomial-GMRES(25,3)
min, max rcond #: 1.92e-17, 7.15e-17
Newton-GMRES(25,3)
min, max rcond #: 2.44e-15, 3.38e-15

6.5 Performance experiments

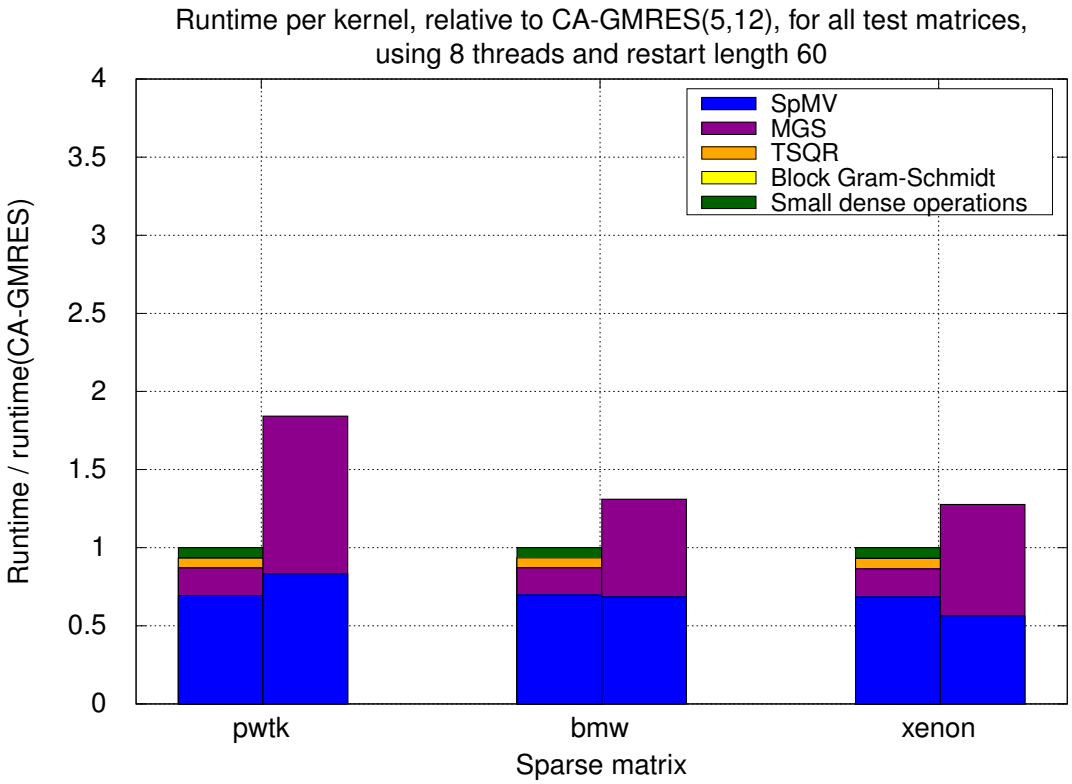


Figure 1: left CA-GMRES, right GMRES

6.5.1 Summary

Conclusion: the MPK is an important kernel and should have been implemented, also restarting with s steps of std. GMRES is not optimal and leaves room for optimization.

7 Conclusion

Krylov subspace methods (summarize this section briefly in introduction)

[1] p.191

- short description:
  Krylov subspace definition: $\mathcal{K}_k(A, r_0) = span\left\{r_0, Ar_0, A^2r_0, \ldots, A^{k-1}r_0\right\}$

All iterative methods build and enhance a KSP with every iteration.

$$\mathbf{r}_k = \mathbf{r}_0 + \sum_{j=1}^{k} c_j A^j \mathbf{r}_0 \quad \rightarrow \quad \mathbf{x}_k = \mathbf{x}_0 + \sum_{j=0}^{k-1} c_{j+1} A^j \mathbf{r}_0$$

what is good for the power method, is bad here, bc. vectors are in theory linearly independent but too close to parallel → in machine arithmetic they become linearly dependent. Need new basis ...

- Arnoldi [1] p.192

$$AQ_k = Q_{k+1} H_{k+1,k}$$

- Summary: ([1] p. 192)
  - 1. construct an orthogonal basis for the Krylov subspace;
  - 2. define an optimality property;
  - 3. use an effective preconditioner.

[1] p.184

- short description: most stable and prominent iterative method, for sym pos def matrices only.
- Algorithm description (just the basics)
  - $\|x - x_k\|_A = min_{y \in \mathcal{K}_k(A,r_0)} \|x - y\|_A$
    After k-steps, x_k minimizes in the KSP the A-norm x - y (only if A is SPD, or else it's not a norm)
  - follows basic concept: $\mathbf{x}_{new} = x_{old} + constant \cdot searchdirection$ (better version *steepest descent*)
  - $\mathbf{r}_k$ is multiple of q_{k+1} (q from Arnoldi; q is not directly used in CG)
    → (1) orthogonal residuals $\mathbf{r}_i^T \mathbf{r}_k = 0, \quad i < k$
    → (2) $(x_i - x_{i-1})^T A (x_k - x_{k-1}) = 0 \quad \rightarrow \quad \Delta \mathbf{x}_i^T A \Delta \mathbf{x}_k = 0, \quad i < k$
    in other words: the corrections in $\mathbf{x}$ are orthogonal in the A-inner product, hence the term 'conjugate' in CG. The term 'gradients' comes from minimizing the energy equation/quadratic form:

$$E(x) = \frac{1}{2} \mathbf{x}^T A \mathbf{x} - b^T \mathbf{x} \rightarrow min$$

    Set the derivative (gradient) to zero, i.e. $E'(x) = 0$:

$$A\mathbf{x} - b = 0$$

    and we're back to the original problem. So minimizing energies and solving the linear equation $A\mathbf{x} = b$ are basically the same.

  - H is symmetric, and therefore tridiagonal. Arnoldi simplifies to Lanczos $\rightarrow$ short 'three-term' recurrences (only have to look at a few previous orthogonal vectors, not all of them).

- p.191 what about general matrices? Any non-singular matrix A can be transformed into SPD matrix via $A^TA \rightarrow$ bad condition number $\kappa(A)$.

  {The condition number of the matrix A^TA is the square of the condition number of A [...] A large condition number both increases the number of iterations required and limits the accuracy to which a solution can be obtained. [4] p. 89 (2.7.40)} consider Krylov subspace methods that are directly based on general matrix $A \rightarrow$ GMRES

GMRES (summarize this section briefly in introduction)

- short description: general form of MINRES (=like CG it is only for symmetric matrices, but must not be PD) MINRES minimizes $\|\mathbf{r}_k\|_2$ and CG minimizes energy norm of the residual $\|\mathbf{r}_k\|_{A^{-1}}$ or the energy norm of the error $\|\mathbf{x}^* - \mathbf{x}_k\|_A$ respectively. [1] (p. 198)

- algorithm description:

- $\|b - A\mathbf{x}_k\|_2 = min_{y \in \mathcal{K}_k(A,r_0)} \|b - Ay\|_2$
  After k-steps, $\mathbf{x}_k$ minimizes in the KSP the ℓ^2-norm $b - Ay$

- GMRES vs. CG:
  - CG forces the residual $\mathbf{r}_k$ to be orthogonal to the Krylov subspace $\mathcal{K}_k(A, r_0)$;
  - GMRES seeks the residual with minimum ℓ_2-norm within the Krylov subspace.

- Summary:
  The main components of a single iteration of GMRES are
  1. perform a step of the Arnoldi process;
  2. update the QR factorization of the updated upper Hessenberg matrix;
  3. solve the resulting least squares problem.

References

[1] Uri M. Ascher and Chen Greif. *A First Course in Numerical Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2011.

[2] Laura Grigori and Sophie Moufawad. Communication avoiding ilu0 preconditioner. *SIAM Journal on Scientific Computing*, 37:C217–C246, 04 2015.

[3] Mark Hoemmen. *Communication-avoiding Krylov Subspace Methods.* PhD thesis, Berkeley, CA, USA, 2010. AAI3413388.

[4] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992.

Appendices

Appendix A

discussion of test matrices