



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, 2019

Studienkennzahl lt. Studienblatt: A 033 521

Fachrichtung: Informatik - Scientific Computing

Betreuerin / Betreuer: Univ.-Prof. Dipl.-Ing. Dr.  
Wilfried Gansterer, M.Sc.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Notation</b>	<b>4</b>
<b>3</b>	<b>Related work</b>	<b>4</b>
<b>4</b>	<b>Computational kernels</b>	<b>5</b>
4.1	Matrix powers kernel (MPK) . . . . .	5
4.2	Tall and skinny QR (TSQR) . . . . .	5
4.3	Block Classical Gram-Schmidt (BCGS) . . . . .	5
<b>5</b>	<b>Arnoldi iteration</b>	<b>5</b>
5.1	Arnoldi(s) . . . . .	6
5.1.1	The Monomial basis . . . . .	6
5.1.2	The Newton basis . . . . .	6
5.2	Arnoldi(s,t) . . . . .	7
5.2.1	Introduction . . . . .	7
5.2.2	Scaling the first basis vector . . . . .	7
5.2.3	QR factorization update . . . . .	7
5.2.4	Reconstructing the upper Hessenberg matrix . . . . .	7
<b>6</b>	<b>CA-GMRES</b>	<b>8</b>
6.1	GMRES(m) . . . . .	8
6.2	Preconditioning . . . . .	10
6.2.1	ILU(0) preconditioner . . . . .	11
6.2.2	CA-ILU(0) preconditioner . . . . .	11
6.3	Convergence metrics . . . . .	11
6.4	Implementation details . . . . .	11
6.5	Numerical experiments . . . . .	12
6.6	Performance experiments . . . . .	16
6.6.1	Summary . . . . .	16
<b>7</b>	<b>Conclusion</b>	<b>16</b>
	<b>Appendices</b>	<b>19</b>

## Abstract

# 1 Introduction

As the CPU-memory performance gap widens the cost for communication increases as well.

Compared to arithmetic costs communication costs are much higher and the widening CPU-memory performance gap promotes the need for communication-avoiding algorithms.

Communication avoiding GMRES is based on s-step GMRES *REF*[[J. Erhel, *A parallel GMRES version for general sparse matrices*, *Electron. Trans. Numer. Anal.*, 3 (1995), pp. 160–176.]]

## 2 Notation

For linear algebra, similar notation is considered as in [3] and [2].

- Greek letters denote scalars, lower case Roman letters denote vectors (or - based on the context - dimensions), capital Roman letters denote matrices.
- Capital letters with two subscripts, e.g. ' $V_{m,n}$ ', denote matrices with  $m$  rows and  $n$  columns.
- Capital *Black letter* letters (e.g.  $V$ ,  $Q$  and  $R$  in Black letters are represented by  $\mathfrak{V}$ ,  $\mathfrak{Q}$  and  $\mathfrak{R}$  resp.) denote matrices that are composed out of other matrices.
- $v_k$  denotes the  $k^{th}$  vector in a series of vectors  $v_0, v_1, \dots, v_k, v_{k+1}, \dots$  of equal length.
- Similarly,  $V_k$  denotes the  $k^{th}$  matrix in a sequence of matrices  $V_0, V_1, \dots, V_k, V_{k+1}, \dots$ . Generally all these matrices have the same number of rows. They may or may not have the same number of columns.
- If  $V_k$  is a matrix consisting of  $s$  vectors  $[v_1, v_2, \dots, v_s]$ ,  $\underline{V}_k$  comprises vectors  $[V_k, v_{s+1}]$ . The underline means one more column at the end.
- If again,  $V_k$  is a matrix consisting of  $s$  vectors  $[v_1, v_2, \dots, v_s]$ ,  $\acute{V}$  consists of vectors  $[v_2, v_3, \dots, v_s]$ . The acute denotes one column less at the beginning.
- As a consequence  $\acute{\underline{V}}$  denotes one more column at the end and one less column at the beginning, e.g.  $\acute{\underline{V}} = [\acute{V}, v_{s+1}]$ .
- Depending on the context, both underline or/and acute letters can also refer to rows as well.
- $0_{m,n}$  is defined as an  $m \times n$  matrix consisting of zeros and  $e_k$  denotes the  $k^{th}$  canonical vector with its dimension depending on the context.
- Matlab notation is used for addressing elements of matrices and vectors. For example, given a matrix  $A$  of size  $n \times n$  and two sets of indices  $\alpha$  and  $\beta$ ,  $A(\alpha, :)$  is a submatrix formed by the subset of the rows of  $A$  whose indices belong to  $\alpha$ . Similarly,  $A(\alpha, \beta)$  is a submatrix formed by the subset of the rows of  $A$  whose indices belong to  $\alpha$  and the subset of the columns of  $A$  whose indices belong to  $\beta$ .

## 3 Related work

s-step methods, CA-ILU(0)

## 4 Computational kernels

In this thesis *computational kernels* define the parts of an algorithm with the highest costs. These costs include both arithmetic operations and communication. The term *communication* generally denotes the movement of data either between different processors or between fast and slow memory. Communication optimal algorithms do not eliminate communication completely, but they are constructed in a way such that reduction of communication is prioritized. This often results in new challenges, e.g., the CA-ILU(0) algorithm (section 6.2.2) has to balance between communication and redundant computations; CA-GMRES (section 6) incorporates additional techniques to deal with ill-conditioned basis vectors.

### 4.1 Matrix powers kernel (MPK)

[3] p.60

The Matrix Powers Kernel Power iteration, SpMV instead of MV, sparse matrix like a graph  $\rightarrow$  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  $\rightarrow$  less communication (factor  $\Theta(s)$  fewer messages) than unblocked CGS. Also, BCGS requires less communication than 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

first presented in REF[[ *W. E. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem, Q. Appl. Maths, 9 (1951), pp. 17-29.*]]

$s$  steps of standard Arnoldi produce an  $s + 1 \times s$  upper Hessenberg Matrix  $\underline{H}$  and  $m \times s + 1$  orthonormal vectors  $q_1, q_2, \dots, q_s, q_{s+1}$  that form a basis for the Krylov subspace  $\mathcal{K}_{s+1}(A, r)$  such that  $AQ = Q\underline{H}$ . There are many ways to orthogonalize successive basis vectors. Modified Gram-Schmidt (MGS) is often employed because it provides high numerical stability, however, although not as stable, Classical Gram-Schmidt (CGS) is more suited for parallel implementations because it provides fewer synchronization points.

introduction.. produces  $AQ = QH$

## 5.1 Arnoldi(s)

[238] H. F. Walker, Implementation of the GMRES and Arnoldi methods using Householder transformations, Tech. Rep. UCRL-93589, Lawrence Livermore National Laboratory, Oct. 1985.

source above! At the heart of Arnoldi's method is a Gram-Schmidt process, and so we refer to the GMRES implementation of [8] as the Gram-Schmidt implementation. The basic form of Arnoldi's method given above employs the classical Gram-Schmidt process, which is numerically untrustworthy. Because of roundoff, there may be severe loss of orthogonality among the computed  $V_m$ 's. In practice, it is usual to implement Arnoldi's method using the modified Gram-Schmidt process (see Golub and Van Loan [4]). Mathematically, this is just a rearrangement of the classical process; computationally, it has superior properties. Even the modified Gram-Schmidt process can fail to perform well if the vectors on which it operates are not sufficiently independent. Indeed, if  $S = (s_1, \dots, s_n)$  is an  $n \times m$  matrix the columns of which are to be orthonormalized and if  $Q = (q_1, \dots, q_m)$  is the computed result of applying modified Gram-Schmidt to the columns of  $S$  using floating point arithmetic with unit rounding error  $u$ , then Bjorck [1] has shown that  $\|Q - QI\|_2 \leq \epsilon$ , where  $\kappa_2(S)$  is the ratio of the largest singular value of  $S$  to the smallest. It follows that at the  $m$ th step of Arnoldi's method using modified Gram-Schmidt,  $V_{m+1}$  may have a significantly nonzero component in the span of  $v_1, \dots, v_m$  if  $\kappa_2((v_1, \dots, v_m, A v_m))$  is large, i. e., if  $A v_m$  is nearly in the span of  $v_1, \dots, v_m$ . Shad [7, p. 214] has suggested that the Gram-Schmidt process in Arnoldi's method may be an important source of errors in the full and incomplete orthogonalization methods ([6], [7]), which are related to GMRES.

### 5.1.1 The Monomial basis

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

### 5.1.2 The Newton basis

polynomial interpolation at shifts  $\theta_1, \theta_2, \dots, \theta_s$ .

**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  $\underline{\mathfrak{Q}}$  which differs from MGS-Arnoldi  $\hat{\underline{\mathfrak{Q}}}$  by a unitary scaling  $\underline{\Theta} = \text{diag}(\theta_1, \theta_2, \dots, \theta_{st}, \theta_{st+1})$  such that  $\hat{\underline{\mathfrak{Q}}} = \underline{\mathfrak{Q}}\underline{\Theta}$ .  $\rightarrow$

- 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.

[3] assumably have some errors in their notation, so the process is outlined slightly differently below.

$$[\underline{\mathfrak{Q}}_0, \underline{\dot{V}}_1] = [\underline{\mathfrak{Q}}_0, \underline{\dot{Q}}_1] \cdot \begin{pmatrix} I_{s+1,s+1} & \underline{\mathfrak{R}}_{0,1} \\ 0_{s,s+1} & \underline{\dot{R}}_1 \end{pmatrix}$$

BGS ...

$$AV_k = \underline{V}_k \underline{B}_k$$

$$A[\underline{\mathfrak{Q}}_{k-1}, V_k] = [\underline{\mathfrak{Q}}_{k-1}, \underline{V}_k] \underline{\mathfrak{B}}_k$$

where  $\underline{\mathfrak{B}}_k$  satisfies:  $\underline{\mathfrak{B}}_k = \begin{pmatrix} \mathfrak{H}_{k-1} & 0_{sk,s} \\ h_{k-1}e_1e_{sk}^T & \underline{B}_k \end{pmatrix}$  with  $\mathfrak{H}_0 := H_0$

$$A[\underline{\mathfrak{Q}}_{k-1}, \underline{\dot{Q}}_k] \cdot \begin{pmatrix} I_{sk+1,sk+1} & \underline{\mathfrak{R}}_{k-1,k} \\ 0_{s-1,sk+1} & \underline{\dot{R}}_k \end{pmatrix} = [\underline{\mathfrak{Q}}_{k-1}, \underline{\dot{Q}}_k] \cdot \begin{pmatrix} I_{sk+1,sk+1} & \underline{\mathfrak{R}}_{k-1,k} \\ 0_{s,sk+1} & \underline{\dot{R}}_k \end{pmatrix} \cdot \begin{pmatrix} \mathfrak{H}_{k-1} & 0_{sk,s} \\ h_{k-1}e_1e_{sk}^T & \underline{B}_k \end{pmatrix}$$

We have

$$A[\underline{\mathfrak{Q}}_{k-1}, \underline{\dot{Q}}_k] = [\underline{\mathfrak{Q}}_{k-1}, \underline{\dot{Q}}_k] \underline{\mathfrak{H}}_k$$

therefore,

$$\underline{\mathfrak{H}}_k = \begin{pmatrix} I_{sk+1,sk+1} & \underline{\mathfrak{R}}_{k-1,k} \\ 0_{s,sk+1} & \underline{\dot{R}}_k \end{pmatrix} \cdot \begin{pmatrix} \mathfrak{H}_{k-1} & 0_{sk,s} \\ h_{k-1}e_1e_{sk}^T & \underline{B}_k \end{pmatrix} \cdot \begin{pmatrix} I_{sk+1,sk+1} & \underline{\mathfrak{R}}_{k-1,k} \\ 0_{s-1,sk+1} & \underline{\dot{R}}_k \end{pmatrix}^{-1}$$

### 5.2.4 Reconstructing the upper Hessenberg matrix

flop optimization. how to apply Givens rotations

## 6 CA-GMRES

### 6.1 GMRES(m)

---

**Algorithm 1** GMRES(m)

---

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

- 1:  $r_0 := b - Ax_0$ ,  $\beta := \|r_0\|_2$ ,  $q_0 := r_0/\beta$ ,  $\underline{Q}_0 := q_0$ ,  $\underline{H}_0 := \emptyset$
- 2: **for**  $k = 1$  to  $m$  **do**
- 3:   Compute  $q_k$  and  $h_k$  using MGS-Arnoldi
- 4:   set  $\underline{Q}_k := [\underline{Q}_{k-1}, q_k]$  and  $\underline{H}_k := [\underline{H}_{k-1}, h_k]$
- 5:   Reduce  $h_k$  of  $\underline{H}_k$  from upper Hessenberg to upper triangular form using  $k$   
     Givens rotations  $G_1, G_2, \dots, G_k$ . Apply the same rotations in the same order  
     to  $\beta e_1$ , resulting in the length  $k+1$  vector  $\zeta_k$ .
- 6:   Element  $k+1$  of  $\zeta_k$  is the 2-norm (in exact arithmetic) of the current residual  
      $r_{k+1} = b - Ax_{k+1}$  of the current solution  $x_{k+1}$ .
- 7:   **if** converged **then**
- 8:     Use the above reduction of  $\underline{H}_k$  to upper triangular form and  $\zeta_k$  to solve  $y_k :=$   
        $\operatorname{argmin}_y \|\underline{H}_k y - \beta e_1\|_2$
- 9:     Set  $x_k := x_0 + \underline{Q}_k y_k$ , and exit
- 10:   **end if**
- 11: **end for**

---

The GMRES method starts with an initial approximate solution  $x_0$  and initial residual  $r_0 = b - Ax_0$  and finds a correction  $z_k$  at iteration  $k$  which solves the least-squares problem

$$z_k := \operatorname{argmin}_z \|b - A(x_0 + z)\|_2 \quad (1)$$

where  $z_k$  is determined in the Krylov subspace

$$\mathcal{K}_k(A, r_0) = \operatorname{span}\{r_0, Ar_0, \dots, A^{k-1}r_0\}.$$

The solution at iteration  $k$  is then formed by  $x_k = x_0 + z_k$ . Since  $\{r_0, Ar_0, \dots, A^{k-1}r_0\}$  is usually ill-conditioned the Arnoldi method is incorporated to produce  $k+1$  orthonormal basis vectors  $\underline{Q} = [q_1, q_2, \dots, q_k, q_{k+1}]$  with  $q_1 = r_0/\|r_0\|_2$  and a  $(k+1) \times k$  upper Hessenberg coefficient matrix  $\underline{H}$  where

$$AQ = \underline{Q}\underline{H}.$$

With these conditions  $z_k$  can be defined as  $z := \underline{Q}y$  such that

$$\begin{aligned} \operatorname{argmin}_z \|b - A(x_0 + z)\|_2 &= \operatorname{argmin}_y \|r_0 - A\underline{Q}y\|_2 \\ &= \operatorname{argmin}_y \|r_0 - \underline{Q}\underline{H}y\|_2. \end{aligned}$$

Since  $q_1 = r_0/\|r_0\|_2$  and  $\underline{Q}$  is orthonormal, one has

$$\begin{aligned} \operatorname{argmin}_y \|r_0 - \underline{Q}\underline{H}y\|_2 &= \operatorname{argmin}_y \|\underline{Q}^T r_0 - \underline{H}y\|_2 \\ &= \operatorname{argmin}_y \|\beta e_1 - \underline{H}y\|_2 \end{aligned} \quad (2)$$



with  $\beta = \|r_0\|_2$ .  $\underline{H}$  is then factored as  $\underline{H} = G\underline{U}$  with  $G$  being a product of  $k$  Givens rotations,  $\underline{U} = \begin{pmatrix} U \\ 0_{1,k} \end{pmatrix}$  and  $U$  being upper triangular. The triangular system to solve is then given by

$$y_k := \operatorname{argmin}_y \|\beta G^T e_1 - \underline{U}y\|_2$$

The solution is obtained by computing  $x_k = x_0 + Qy_k$ . Note that the absolute value of the last coordinate of  $\beta G^T e_1$  is  $\|b - Ax_k\|_2$ , the absolute residual at iteration  $k$ .

The CA-GMRES algorithm solves a different least-squares problem than (2):

$$\operatorname{argmin}_y \|\beta e_1 - \underline{R}\underline{B}R^{-1}y\|_2 \tag{3}$$

---

**Algorithm 2** Newton CA-GMRES

---

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

```
1:  $r_0 := b - Ax_0$ ,  $\beta := \|r_0\|_2$ ,  $q_1 := r_0/\beta$ 
2: for  $k = 0$  to  $t - 1$  do
3:   if  $k = 0$  then
4:     Compute  $\underline{Q}_0$  and  $\underline{H}_0$  using MGS-Arnoldi
5:     Set  $\underline{\Omega}_0 := \underline{Q}_0$  and  $\underline{\mathfrak{H}}_0 := \underline{H}_0$ 
6:     Compute Ritz values from  $H_0$  and fix basis conversion matrix  $\underline{B}_k$ 
7:     Reduce  $\underline{H}_0$  from upper Hessenberg to upper triangular form using  $s$  Givens
        rotations  $G_1, G_2, \dots, G_s$ . Apply the same rotations in the same order to  $\beta e_1$ ,
        resulting in the length  $s + 1$  vector  $\zeta_0$ .
8:   else
9:     Compute  $\underline{V}_k$  using SpMV and 1-2 AXPY
10:     $\underline{\mathfrak{R}}_{k-1,k} := \underline{\Omega}_{k-1}^T \underline{V}_k$ 
11:     $\underline{V}'_k := \underline{V}_k - \underline{\Omega}_{k-1} \underline{\mathfrak{R}}_{k-1,k}$ 
12:    Compute QR factorization of  $\underline{V}'_k \rightarrow \underline{Q}_k \underline{R}_k$  using TSQR
13:    Compute  $\underline{\mathfrak{H}}_{k-1,1} := -\mathfrak{H}_{k-1} \mathfrak{R}_{k-1,k} R_k^{-1} + \mathfrak{R}_{k-1,k} \underline{B}_k R_k^{-1}$ 
14:    Compute  $\underline{H}_k := R_k B_k R_k^{-1} + \tilde{\rho}_k^{-1} b_k z_k e_s^T - h_{k-1} e_1 e_{s(k-1)}^T \mathfrak{R}_{k-1,k} R_k^{-1}$ 
15:    Compute  $h_k := \tilde{\rho}_k^{-1} \rho_k b_k$ 
16:     $\underline{\mathfrak{H}}_k := \begin{pmatrix} \mathfrak{H}_{k-1} & \underline{\mathfrak{H}}_{k-1,k} \\ h_0 e_1 e_{sk}^T & \underline{H}_k \\ 0_{1,sk} & h_k e_s^T \end{pmatrix}$ 
17:    Apply Givens rotations  $G_1, \dots, G_{sk}$  in order to  $\begin{pmatrix} \underline{\mathfrak{H}}_{k-1,k} \\ \underline{H}_k \end{pmatrix}$ .
18:    Reduce  $\underline{H}_k$  to upper triangular form using  $s$  Givens rotations  $G_{sk+1}, \dots, G_{s(k+1)}$ .
        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  $\zeta_k$ .
19:   end if
20:   Element  $s(k+1)+1$  of  $\zeta_k$  is the 2-norm (in exact arithmetic) of the current residual
         $r_{k+1} = b - Ax_{k+1}$  of the current solution  $x_{k+1}$ .
21:   if converged then
22:     Use the above reduction of  $\underline{\mathfrak{H}}_k$  to upper triangular form and  $\zeta_k$  to solve  $y_k :=$ 
         $\operatorname{argmin}_y \|\underline{\mathfrak{H}}_k y - \beta e_1\|_2$ 
23:     Set  $x_k := x_0 + \underline{\Omega}_k y_k$ , and exit
24:   end if
25: end for
```

---

## 6.2 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  $A' = D_rAD_c$  with  $D_r$  and  $D_c$  diagonal.

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.

### 6.2.1 ILU(0) preconditioner

$M = LU$

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

### 6.2.2 CA-ILU(0) preconditioner

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

## 6.3 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  $A$ .

## 6.4 Implementation details

language: C++, libraries: intel MKL,

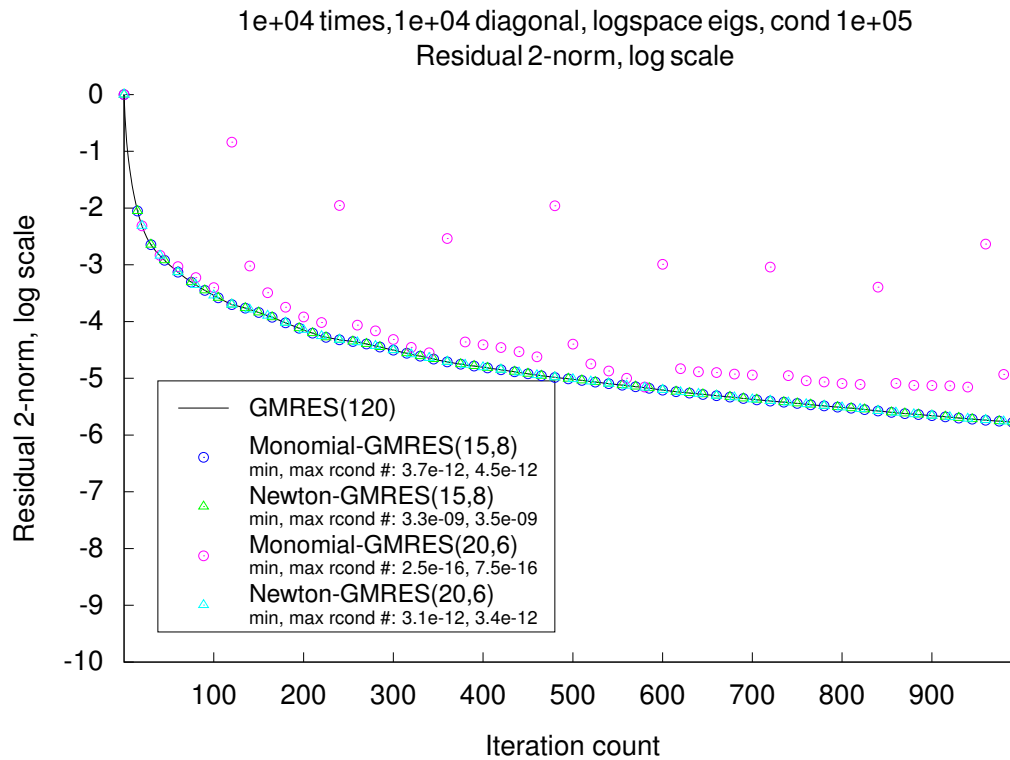
The Intel® Math Kernel Library has been optimized by exploiting both processor and system features and capabilities. Special care has been given to those routines that most profit from cache-management techniques. These especially include matrix-matrix operation routines such as `dgemm()`. In addition, code optimization techniques have been applied to minimize dependencies of scheduling integer and floating-point units on the results within the processor. The major optimization techniques used throughout the library include: • Loop unrolling to minimize loop management costs • Blocking of data to improve data reuse opportunities • Copying to reduce chances of data eviction from cache • Data prefetching to help hide memory latency • Multiple simultaneous operations (for example, dot products in `dgemm`) to eliminate stalls due to arithmetic unit pipelines

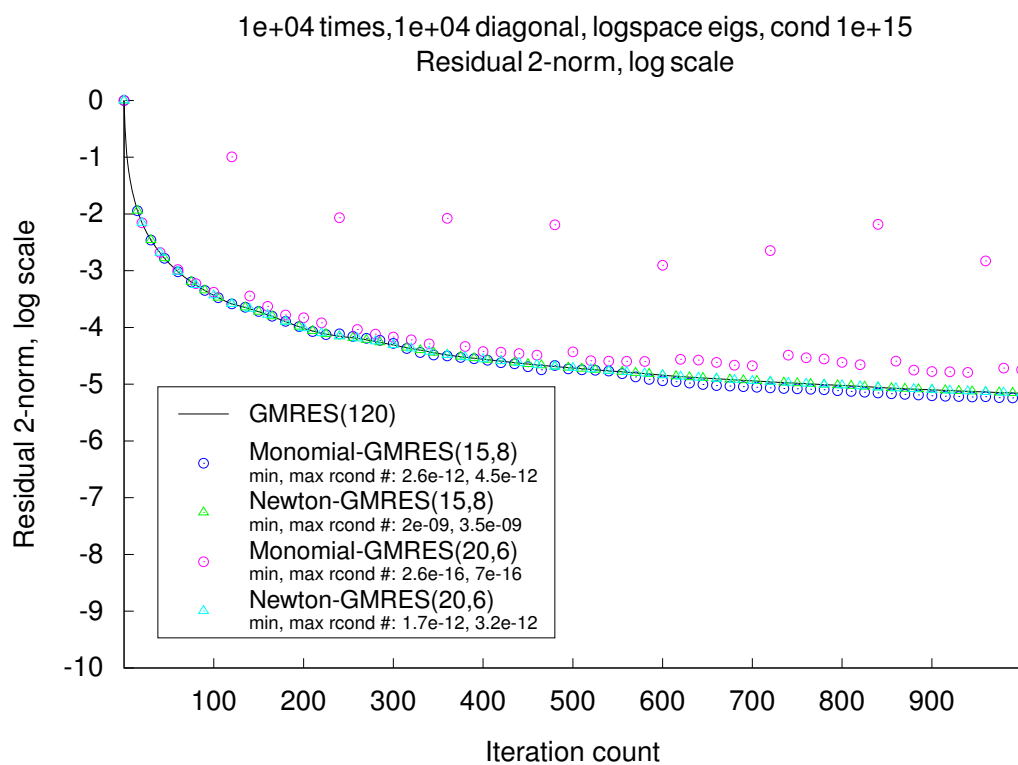
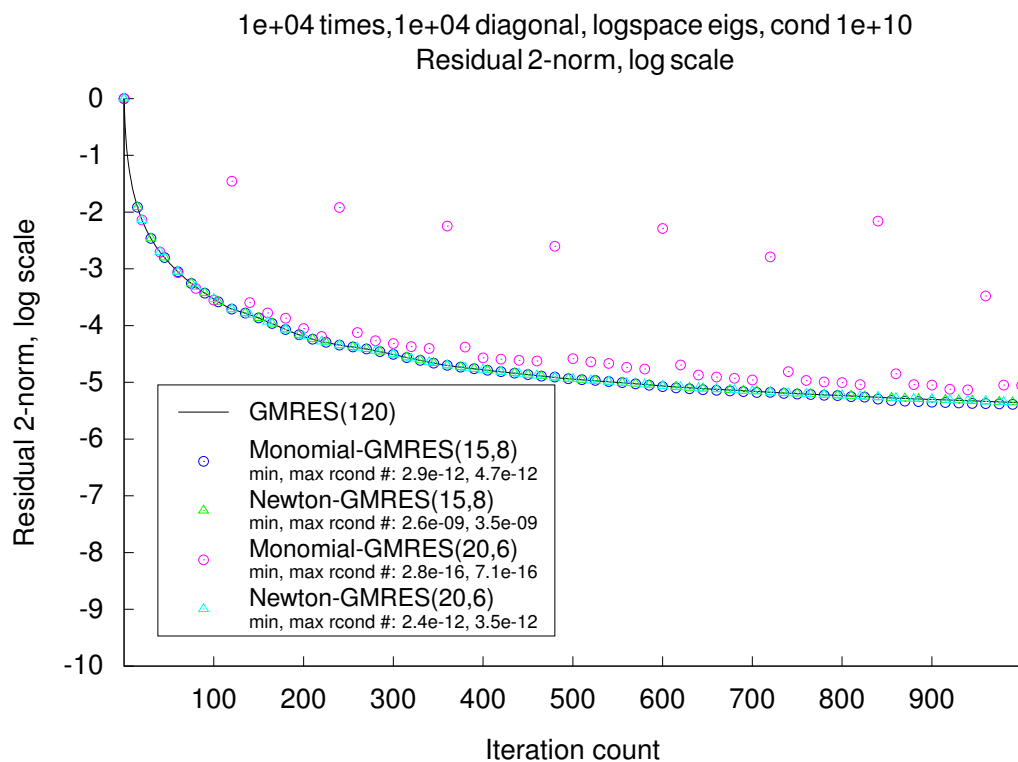
- Use of hardware features such as the SIMD arithmetic units, where appropriate These are techniques from which the arithmetic code benefits the most.

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].

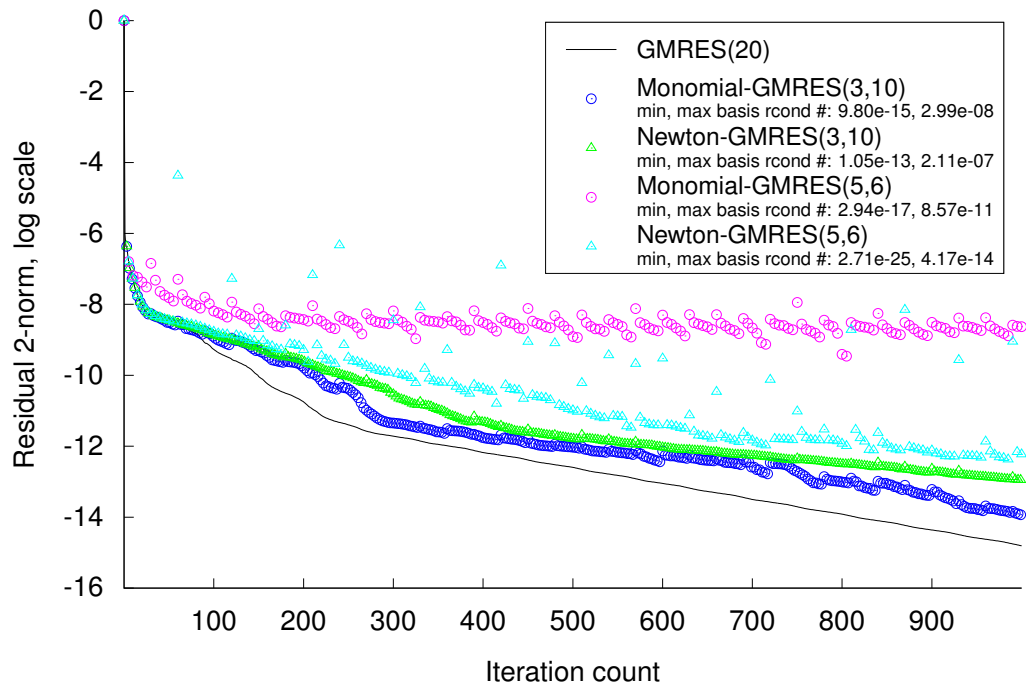
## 6.5 Numerical experiments

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).

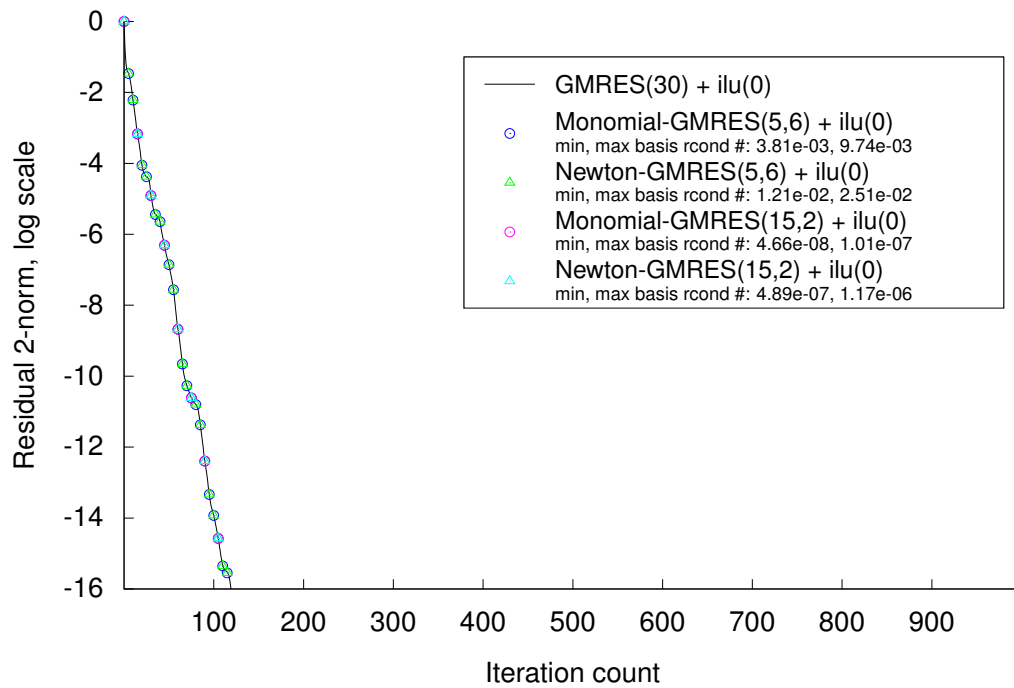


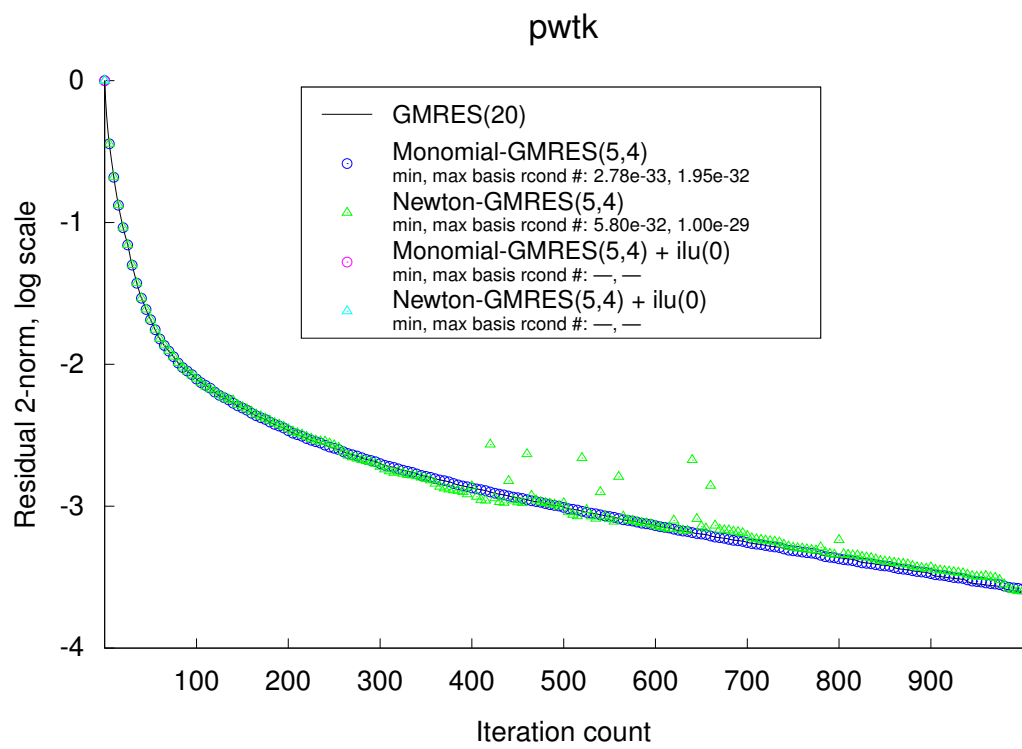


### Watt1



### Watt1





## 6.6 Performance experiments

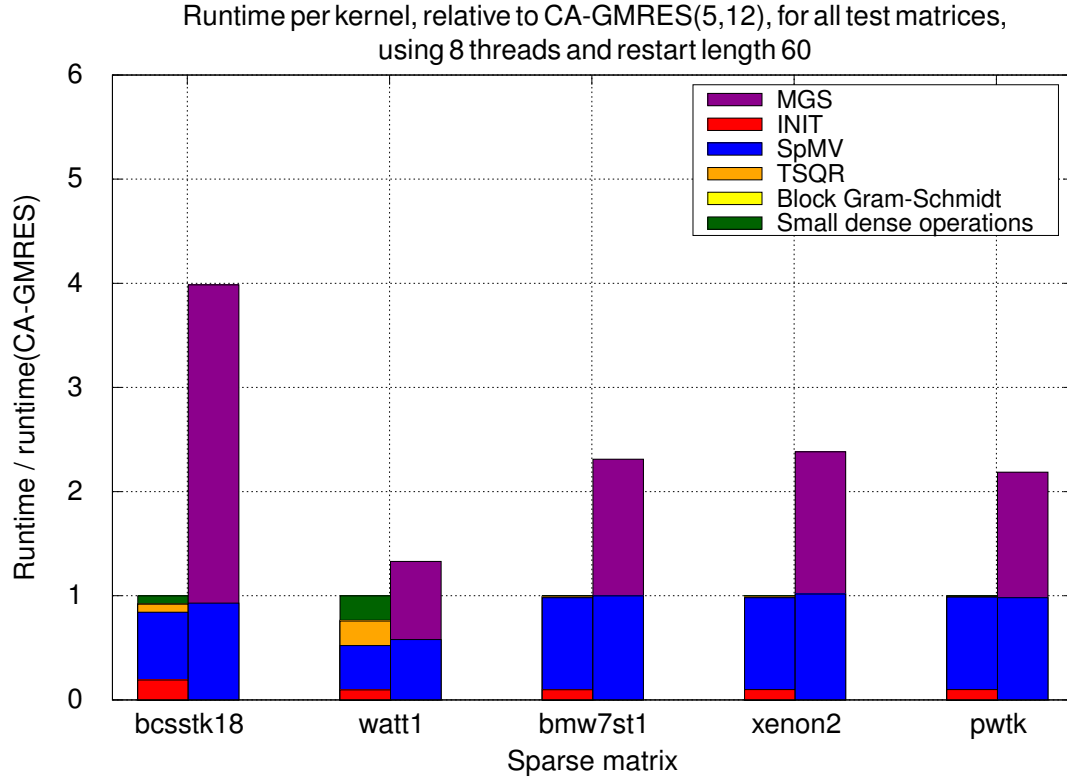


Figure 1: left CA-GMRES, right GMRES

### 6.6.1 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:  $\mathcal{K}_k(A, r_0) = \text{span} \{r_0, Ar_0, A^2r_0, \dots, A^{k-1}r_0\}$



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  $\rightarrow$  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)
  - $\|x - x_k\|_A = \operatorname{argmin}_{y \in \mathcal{K}_k(A, r_0)} \|x - y\|_A$   
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)
    - $\rightarrow$  (1) orthogonal residuals  $\mathbf{r}_i^T \mathbf{r}_k = 0, \quad i < k$
    - $\rightarrow$  (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  $A^T A \rightarrow$  bad condition number  $\kappa(A)$ .

{The condition number of the matrix  $A^T A$  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:
- $\|b - A\mathbf{x}_k\|_2 = \operatorname{argmin}_{y \in \mathcal{K}_k(A, r_0)} \|b - Ay\|_2$   
After  $k$ -steps,  $\mathbf{x}_k$  minimizes in the KSP the  $\ell^2$ -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  $\ell_2$ -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.

## List of Algorithms

1	GMRES(m) . . . . .	8
2	Newton CA-GMRES . . . . .	10

## Appendices

### Appendix A

discussion of test matrices