# S-Step and Communication-Avoiding Iterative Methods

Maxim Naumov
NVIDIA, 2701 San Tomas Expressway, Santa Clara, CA 95050

**Abstract**

In this paper we make an overview of s-step Conjugate Gradient and develop a novel formulation for s-step BiConjugate Gradient Stabilized iterative method. Also, we show how to add preconditioning to both of these s-step schemes. We explain their relationship to the standard, block and communication-avoiding counterparts. Finally, we explore their advantages, such as the availability of matrix-power kernel $A^k\mathbf{x}$ and use of block-dot products $B = X^TY$ that group individual dot products together, as well as their drawbacks, such as the extra computational overhead and numerical stability related to the use of monomial basis for Krylov subspace $\mathbb{K}_s = \{\mathbf{r}, A\mathbf{r}, ..., A^{s-1}\mathbf{r}\}$. We note that the mathematical techniques used in this paper can be applied to other methods in sparse linear algebra and related fields, such as optimization.

## 1 Introduction

In this paper we are concerned with investigating iterative methods for the solution of linear system

$$A\mathbf{x} = \mathbf{f} \tag{1}$$

where nonsingular coefficient matrix $A \in \mathbb{R}^{n\times n}$, right-hand-side (RHS) $\mathbf{f}$ and solution $\mathbf{x} \in \mathbb{R}^n$. A comprehensive overview of standard iterative methods can be found in [1, 17]. In this brief study we will focus on two popular algorithms: (i) Conjugate Gradient (CG) and (ii) BiConjugate Gradient Stabilized (BiCGStab) Krylov subspace iterative methods. These algorithms are often used to solve linear systems with symmetric positive definite (SPD) and nonsymmetric coefficient matrices, respectively.

---

NVIDIA Technical Report NVR-2016-003, April 2016.


This research was developed, in part, with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions, and findings contained in this article are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

The CG and BiCGStab methods have already been generalized to their block counterparts in [11, 15]. The block methods work with $s$ vectors simultaneously and were originally designed to solve linear systems with multiple RHS, such as

$$AX = F \tag{2}$$

where multiple RHS and solutions are column vectors in tall matrices $F$ and $X \in \mathbb{R}^{n\times s}$, respectively. However, they can be adapted to solve systems with a single RHS, by for example randomly selecting the remaining $s-1$ RHS [14]. In the latter case, the block methods often perform less iterations than their standard counterparts, but they do not compute exactly the same solution in $i$ iterations as the standard methods in $i\times s$ iterations.

The s-step methods occupy a middle ground between their standard and block counterparts. They are usually applied to linear systems with a single RHS, they work with $s$ vectors simultaneously, but they are also designed to produce identical solution in $i$ iterations as the standard methods in $i \times s$ iterations, when the computation is performed in exact arithmetic. In fact, the s-step methods can be viewed as an unrolling of $s$ iterations and a clever re-grouping of recurrence terms in them.

The s-step methods were first investigated by J. Van Rosendale, A. T. Chronopoulos and C. W. Gear in [4, 6, 16], where the authors proposed s-step CG algorithm for SPD linear systems. Subsequent work by A. T. Chronopoulos and C. D. Swanson in [5, 7] lead to the development of s-step methods, such as GMRES and Orthomin, but not BiCGStab, for nonsymmetric linear systems. These methods were further improved and generalized to the solution of eigenvalue problems using Lanczos and Arnoldi iterations in [8, 9].

There are several advantages for using s-step methods. The first advantage is that the matrix-vector multiplications used to build the Krylov subspace $\mathbb{K}_s = \{\mathbf{r}, A\mathbf{r}, ..., A^{s-1}\mathbf{r}\}$ can be performed together, one immediately after another. This allows us to develop a more efficient matrix-vector multiplication, using a pipeline where the results from current multiplication are immediately forwarded as inputs to the next. The second advantage is that the dot products spread across multiple iterations are bundled together. This allows us to minimize fan-in communication that is associated with dot-products and often limits scalability of parallel platforms.

There are also some disadvantages for using s-step methods. We perform more work per 1 iteration of an s-step method than per $s$ iterations of a standard method. The extra work is often the result of one extra matrix-vector multiplication required to recompute the residual. Also, notice that in practice the vectors computed by matrix-power kernel $A^k\mathbf{x}$ for $k = 0, ..., s-1$ quickly become linearly dependent. Therefore, s-step methods can suffer from issues related to numerical stability resulting from the use of a monomial basis for the Krylov subspace corresponding to $s$ internal iterations.

In this paper we first make an overview of how to use properties of the directions and residual vectors in standard CG to build the s-step CG iterative method. Then, we find the properties of the directions and residual vectors in standard BiCGStab. Finally, we use this information in conjuction with recurrence relationships for these vectors to develop a novel s-step BiCGStab iterative method. We also show how to add preconditioning to both of these s-step schemes.

Finally, we point out that s-step iterative methods are closely related to communication-avoiding algorithms proposed in [2, 3, 10, 12]. In fact, the main differences between these methods lie in (i) a different basis used by the communication-avoiding algorithms to address numerical stability, and (ii) an efficient (communication-avoiding) implementation of matrix-power kernel $A^k\mathbf{x}$ as well as distributed dense linear algebra operations, such as dense QR-factorization used in GMRES [18].

Let us now show how to derive the well known s-step CG and novel s-step BiCGStab iterative methods as well as how to add preconditioning to both algorithms.

## 2 S-Step Conjugate Gradient (CG)

The standard CG method is shown in Alg. 1.

**Algorithm 1** Standard CG

1: Let $A \in \mathbb{R}^{n\times n}$ be a SPD matrix and $\mathbf{x}_0$ be an initial guess.
2: Compute $\mathbf{r}_0 = \mathbf{f} - A\mathbf{x}_0$
3: **for** $i = 0, 1, ...$until convergence **do**
4: **if** $i == 0$ **then** ▷ Find directions $\mathbf{p}$
5: Set $\mathbf{p}_i = \mathbf{r}_i$
6: **else**
7: Compute $\beta_i = \frac{\mathbf{r}_i^T\mathbf{r}_i}{\mathbf{r}_{i-1}^T\mathbf{r}_{i-1}}$ ▷ Dot-product
8: Compute $\mathbf{p}_i = \mathbf{r}_i + \beta_i\mathbf{p}_{i-1}$
9: **end if**
10: Compute $\mathbf{q}_i = A\mathbf{p}_i$ ▷ Matrix-vector multiply
11: Compute $\alpha_i = \frac{\mathbf{r}_i^T\mathbf{r}_i}{\mathbf{r}_i^T\mathbf{q}_i}$ ▷ Dot-product
12: Compute $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i\mathbf{p}_i$
13: Compute $\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i\mathbf{q}_i$
14: **if** $||\mathbf{r}_{i+1}||_2/||\mathbf{r}_0||_2 \leq$ tol **then** stop **end** ▷ Check Convergence
15: **end for**

Let us now prove that the directions $\mathbf{p}_i$ and residuals $\mathbf{r}_i$ computed by CG satisfy a few properties. These properties will subsequently be used to setup the s-step CG method.

**Lemma 1.** *The residuals $\boldsymbol{r}_i$ are orthogonal and directions $\boldsymbol{p}_i$ are A-orthogonal*

$$\boldsymbol{r}_i^T\boldsymbol{r}_j = 0 \quad \textit{and} \quad \boldsymbol{p}_i^T A\boldsymbol{p}_j = 0 \quad \textit{for} \quad i \neq j \tag{3}$$

*Proof.* See Proposition 6.13 in [17]. □

**Theorem 1.** *The current residual $\boldsymbol{r}_i$ and all previous direction $\boldsymbol{p}_j$ are orthogonal*

$$\boldsymbol{r}_i^T\boldsymbol{p}_j = 0 \textit{ for } i > j \tag{4}$$

*Proof.* Using induction. Notice that for the base case $i = 1$ we have

$$\mathbf{r}_1^T \mathbf{p}_0 = \mathbf{r}_1^T \mathbf{r}_0 = 0 \tag{5}$$

For the induction step, let us assume $\mathbf{r}_i^T \mathbf{p}_{j-1} = 0$. Then

$$\mathbf{r}_i^T \mathbf{p}_j \quad = \quad \mathbf{r}_i^T (\mathbf{r}_j + \beta_{j-1} \mathbf{p}_{j-1}) = \beta_{j-1} \mathbf{r}_i^T \mathbf{p}_{j-1} = 0 \tag{6}$$

□

Let us now derive an alternative expression for scalars $\alpha_i$ and $\beta_i$.

**Corollary 1.** *The scalars $\alpha_i$ and $\beta_i$ satisfy*

$$(\boldsymbol{p}_i^T A \boldsymbol{p}_i)\alpha_i \quad = \quad \boldsymbol{p}_i^T \boldsymbol{r}_i \tag{7}$$

$$(\boldsymbol{p}_i^T A \boldsymbol{p}_i)\beta_i \quad = \quad -\boldsymbol{p}_i^T A \boldsymbol{r}_{i+1} \tag{8}$$

*Proof.* Using Lemma 1 and Theorem 1 we have

$$\begin{aligned} \mathbf{r}_i^T \mathbf{r}_{i+1} &= \mathbf{r}_i^T (\mathbf{r}_i - \alpha_i A \mathbf{p}_i) = (\mathbf{p}_i - \beta_{i-1} \mathbf{p}_{i-1})^T \mathbf{r}_i - \alpha_i (\mathbf{p}_i - \beta_{i-1} \mathbf{p}_{i-1})^T A \mathbf{p}_i \\ &= \mathbf{p}_i^T \mathbf{r}_i - \alpha_i \mathbf{p}_i^T A \mathbf{p}_i = 0 \end{aligned} \tag{9}$$

and

$$\mathbf{p}_i^T A \mathbf{p}_{i+1} \quad = \quad \mathbf{p}_i^T A (\mathbf{r}_{i+1} + \beta_i A \mathbf{p}_i) = \mathbf{p}_i^T A \mathbf{r}_{i+1} + \beta_i \mathbf{p}_i^T A \mathbf{p}_i = 0 \tag{10}$$

□

Let us now assume that we have $s$ directions $\mathbf{p}_i$ available at once, then we can write

$$\mathbf{x}_{i+s} = \mathbf{x}_i + \alpha_i \mathbf{p}_i + ... + \alpha_{i+s-1} \mathbf{p}_{i+s-1} = \mathbf{x}_i + P_i \mathbf{a}_i \tag{11}$$

where $P_i = [\mathbf{p}_i, ..., \mathbf{p}_{i+s-1}] \in \mathbb{R}^{n \times s}$ is a tall matrix and $\mathbf{a}_i = [\alpha_i, ..., \alpha_{i+s-1}]^T \in \mathbb{R}^s$ is a small vector. Consequently, residual can be expressed as

$$\mathbf{r}_{i+s} = \mathbf{f} - A \mathbf{x}_{i+s} = \mathbf{r}_i - A P_i \mathbf{a}_i \tag{12}$$

Also, let us assume that we have the monomial basis for Krylov subspace for $s$ iterations

$$R_i = [\mathbf{r}_i, A\mathbf{r}_i, ..., A^{s-1} \mathbf{r}_i] \tag{13}$$

Then, following the standard CG algorithm, it would be natural to express the next directions as a combination of previous directions and the basis vectors

$$P_{i+1} = R_{i+1} + P_i B_i \tag{14}$$

where $B_i = [\beta_{i,j}] \in \mathbb{R}^{s \times s}$ is a small matrix. At this point we have almost all the building blocks of s-step CG method, but we are still missing a way to find scalars $\alpha$ and $\beta$ stored in the vector $\mathbf{a}_i$ and matrix $B_i$. These coefficients can be found using the following Corollary.

**Corollary 2.** *The vector* $\boldsymbol{a}_i^T = [\alpha_i, ..., \alpha_{i+s-1}]$ *and matrix* $B_i = [\beta_{i,j}]$ *satisfy*

$$(P_i^T A P_i)\boldsymbol{a}_i = P_i^T \boldsymbol{r}_i \tag{15}$$

$$(P_i^T A P_i)B_i = -P_i^T A R_{i+1} \tag{16}$$

*Proof.* Enforcing orthogonality of residual and search directions in Theorem 1 we have

$$P_i^T \mathbf{r}_{i+s} = P_i^T(\mathbf{r}_i - AP_i\mathbf{a}_i) = P_i^T\mathbf{r}_i - (P_i^T AP_i)\mathbf{a}_i = 0 \tag{17}$$

Also, enforcing $A$-orthogonality between search directions in Lemma 1 we obtain

$$P_i^T A P_{i+1} = P_i^T A(R_{i+1} + P_i B_i) = P_i^T A R_{i+1} + (P_i^T AP_i)B_i = 0 \tag{18}$$

□

Notice the similarity between Corollary 1 and 2 for standard and s-step CG, respectively.

Finally, putting all the equations (11) - (16) together we obtain the pseudo-code for s-step CG iterative method, shown in Alg. 2.

**Algorithm 2** S-Step CG

Let $A \in \mathbb{R}^{n\times n}$ be a SPD matrix and $\mathbf{x}_0$ be an initial guess.
Compute $\mathbf{r}_0 = \mathbf{f} - A\mathbf{x}_0$ and let index $k = is$ (initially $k = 0$)
**for** $i = 0, 1, ...$until convergence **do**
    Compute $T = [\mathbf{r}_k, A\mathbf{r}_k, ..., A^s\mathbf{r}_k]$ ▷ Matrix-power kernel
    Let $R_i = [\mathbf{r}_k, A\mathbf{r}_k, ..., A^{s-1}\mathbf{r}_k]$ ▷ Extract $R = T(:, 1:s-1)$ from $T$
    Let $Q_i = [A\mathbf{r}_k, A^2\mathbf{r}_k, ..., A^s\mathbf{r}_k] = AR_i$ ▷ Extract $Q = T(:, 2:s)$ from $T$
    **if** $i == 0$ **then** ▷ Find directions $\mathbf{p}$
        Set $P_i = R_i$
    **else**
        Compute $C_i = -Q_i^T P_{i-1}$ ▷ Block dot-products
        Solve $W_{i-1}B_i = C_i$ ▷ Find scalars $\beta$
        Compute $P_i = R_i + P_i B_i$
    **end if**
    Compute $W_i = Q_i^T P_i$ ▷ Block dot-products
    Compute $\mathbf{g}_i = P_i^T \mathbf{r}_i$
    Solve $W_i\mathbf{a}_i = \mathbf{g}_i$ ▷ Find scalars $\alpha$
    Compute $\mathbf{x}_{k+s} = \mathbf{x}_k + P_i\mathbf{a}_i$ ▷ Compute new approximation $\mathbf{x}_k$
    Compute $\mathbf{r}_{k+s} = \mathbf{f} - A\mathbf{x}_{k+s}$ ▷ Recompute residual $\mathbf{r}_k$
    **if** $||\mathbf{r}_{k+s}||_2/||\mathbf{r}_0||_2 \leq$ tol **then** stop **end** ▷ Check Convergence
**end for**

Let us now introduce preconditioning into the s-step CG in a manner similar to the standard algorithm. Let the preconditioned linear system be

$$(L^{-1}AL^{-T})(L^T\mathbf{x}) = L^{-1}\mathbf{f} \tag{19}$$

Then, we can apply the s-step on the preconditioned system (19) and re-factor recurrences in terms of preconditioner $M = LL^T$. If we express everything in terms of "preconditioned" directions $M^{-1}P_i$ and work with the original solution vector $\mathbf{x}_i$ we obtain the preconditioned s-step CG method, shown in Alg. 3.

Notice that tall matrices $Z_i$ and $Q_i$ on lines 6 and 7 can be constructed during the computation of $T$ on line 5 in Alg. 3. The key observation is that the computation of $T$ proceeds in the following fashion

$$\mathbf{r}_i, M^{-1}\mathbf{r}_i, (AM^{-1})\mathbf{r}_i, M^{-1}(AM^{-1})\mathbf{r}_i, (AM^{-1})^2\mathbf{r}_i, ..., M^{-1}(AM^{-1})^{s-1}\mathbf{r}_i, (AM^{-1})^s\mathbf{r}_i$$

and therefore alternating odd and even terms define columns of tall matrices $Z_i$ and $Q_i$.

**Algorithm 3** Preconditioned S-Step CG

1: Let $A \in \mathbb{R}^{n\times n}$ be a SPD matrix and $\mathbf{x}_0$ be an initial guess.
2: Let $M = LL^T$ be SPD preconditioner, so that $M^{-1} \approx A^{-1}$.
3: Compute $\mathbf{r}_0 = \mathbf{f} - A\mathbf{x}_0$, $\hat{\mathbf{r}}_0 = M^{-1}\mathbf{r}_0$ and let index $k = is$ (initially $k = 0$)
4: **for** $i = 0, 1, ...$until convergence **do**
5: Compute $T = M^{-1}[\mathbf{r}_k, \hat{A}\mathbf{r}_k, ..., \hat{A}^s\mathbf{r}_k]$ ▷ Matrix-power kernel
6: Let $Z_i = M^{-1}[\mathbf{r}_k, \hat{A}\mathbf{r}_k, ..., \hat{A}^{s-1}\mathbf{r}_k] = M^{-1}R_i$ ▷ Build $Z$ & $Q$ during
7: Let $Q_i = [\hat{A}\mathbf{r}_k, \hat{A}^2\mathbf{r}_k, ..., \hat{A}^s\mathbf{r}_k] = AZ_i$ ▷ computation of $T$
8: where auxiliary $\hat{A} = AM^{-1}$ and $R_i = [\mathbf{r}_k, \hat{A}\mathbf{r}_k, ..., \hat{A}^{s-1}\mathbf{r}_k]$
9: **if** $i == 0$ **then** ▷ Find directions $\mathbf{p}$
10: Set $P_i = Z_i$
11: **else**
12: Compute $C_i = -Q_i^T P_{i-1}$ ▷ Block dot-products
13: Solve $W_{i-1}B_i = C_i$ ▷ Find scalars $\beta$
14: Compute $P_i = Z_i + P_iB_i$
15: **end if**
16: Compute $W_i = Q_i^T P_i$ ▷ Block dot-products
17: Compute $\mathbf{g}_i = P_i^T\mathbf{r}_i$
18: Solve $W_i\mathbf{a}_i = \mathbf{g}_i$ ▷ Find scalars $\alpha$
19: Compute $\mathbf{x}_{k+s} = \mathbf{x}_k + P_i\mathbf{a}_i$ ▷ Compute new approximation $\mathbf{x}_k$
20: Compute $\mathbf{r}_{k+s} = \mathbf{f} - A\mathbf{x}_{k+s}$ ▷ Recompute residual $\mathbf{r}_k$
21: Compute $\hat{\mathbf{r}}_{k+s} = M^{-1}\mathbf{r}_{k+s}$
22: **if** $||\mathbf{r}_{k+s}||_2/||\mathbf{r}_0||_2 \leq$ tol **then** stop **end** ▷ Check Convergence
23: **end for**

Finally, notice that preconditioned s-step CG performs an extra matrix-vector multiplication with $A$ and preconditioner solve with $M^{-1}$ per 1 iteration, when compared with $s$ iterations of the standard preconditioned CG method. The extra work happens during re-computation of residual at the end of every s-step iteration.

# 3 S-Step BiConjugate Gradient Stabilized (BiCGStab)

Let us now focus on the s-step BiCGStab method, which has not been previously derived by A. T. Chronopoulos et al. The standard BiCGStab is given in Alg 4.

**Algorithm 4** Standard BiCGStab

1: Let $A \in \mathbb{R}^{n \times n}$ be a nonsingular matrix and $\mathbf{x}_0$ be an initial guess.
2: Compute $\mathbf{r}_0 = \mathbf{f} - A\mathbf{x}_0$ and set $\mathbf{p}_0 = \mathbf{r}_0$
3: Let $\check{\mathbf{r}}_0$ be arbitrary (for example $\check{\mathbf{r}}_0 = \mathbf{r}_0$)
4: **for** $i = 0, 1, ...$until convergence **do**
5: Compute $\mathbf{z}_i = A\mathbf{p}_i$ ▷ Matrix-vector multiply
6: Compute $\alpha_i = \frac{\check{\mathbf{r}}_0^T \mathbf{r}_i}{\check{\mathbf{r}}_0^T \mathbf{z}_i}$ ▷ Dot-product
7: Compute $\mathbf{s}_i = \mathbf{r}_i - \alpha_i \mathbf{z}_i$ ▷ Find directions $\mathbf{s}$
8: **if** $||\mathbf{s}_i||_2 / ||\mathbf{r}_0||_2 \leq$ tol **then** set $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i$ and stop **end** ▷ Early exit
9: Compute $\mathbf{v}_i = A\mathbf{s}_i$ ▷ Matrix-vector multiply
10: Compute $\omega_i = \frac{\mathbf{s}_i^T \mathbf{v}_i}{\mathbf{v}_i^T \mathbf{v}_i}$ ▷ Dot-product
11: Compute $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i + \omega_i \mathbf{s}_i$
12: Compute $\mathbf{r}_{i+1} = \mathbf{s}_i - \omega_i \mathbf{v}_i$
13: **if** $||\mathbf{r}_{i+1}||_2 / ||\mathbf{r}_0||_2 \leq$ tol **or** early exit **then** stop **end** ▷ Check Convergence
14: Compute $\beta_i = \left(\frac{\check{\mathbf{r}}_0^T \mathbf{r}_{i+1}}{\check{\mathbf{r}}_0^T \mathbf{r}_i}\right)\left(\frac{\alpha_i}{\omega_i}\right)$
15: Compute $\mathbf{p}_{i+1} = \mathbf{r}_{i+1} + \beta_i(\mathbf{p}_i - \omega_i \mathbf{z}_i)$ ▷ Find directions $\mathbf{p}$
16: **end for**

Recall that the directions $\mathbf{p}_i$ and residuals $\mathbf{r}_i$ in BiCGStab have been derived from BiCG method using polynomial recurrences to avoid multiplication with transpose of the coefficient matrix $A^T$ [17]. The properties of directions and residuals in BiCG method are well known and can be easily used to setup the corresponding s-step method. The relationships between them in BiCGStab are far less clear. Therefore, the first step is to understand the properties of directions $\mathbf{s}_i$, $\mathbf{p}_i$ and residuals $\mathbf{r}_i$, $\check{\mathbf{r}}_0$ in BiCGStab method.

**Theorem 2.** *The directions $\boldsymbol{s}_i$, $\boldsymbol{p}_i$ and residuals $\boldsymbol{r}_i$, $\check{\boldsymbol{r}}_0$ satisfy the following relationships*

$$\check{\boldsymbol{r}}_0^T \boldsymbol{s}_i = 0 \tag{20}$$

$$\boldsymbol{r}_{i+1}^T A \boldsymbol{s}_i = 0 \tag{21}$$

$$\check{\boldsymbol{r}}_0^T (\boldsymbol{p}_{i+1} - \beta_i \boldsymbol{p}_i) = 0 \tag{22}$$

*Proof.* First relationship (20) follows from

$$\check{\mathbf{r}}_0^T \mathbf{s}_i = \check{\mathbf{r}}_0^T (\mathbf{r}_i - \alpha_i A\mathbf{p}_i) = \check{\mathbf{r}}_0^T \mathbf{r}_i - \alpha_i \check{\mathbf{r}}_0^T A\mathbf{p}_i = \check{\mathbf{r}}_0^T \mathbf{r}_i - \check{\mathbf{r}}_0^T \mathbf{r}_i = 0 \tag{23}$$

Second relationship (21) follows from

$$\mathbf{r}_{i+1}^T A\mathbf{s}_i = (\mathbf{s}_i - \omega_i A\mathbf{s}_i)^T A\mathbf{s}_i = \mathbf{s}_i^T A\mathbf{s}_i - \omega_i (A\mathbf{s}_i)^T (A\mathbf{s}_i) = \mathbf{s}_i^T A\mathbf{s}_i - \mathbf{s}_i^T A\mathbf{s}_i = 0 \tag{24}$$

The last relationship (22) follows from

$$\begin{aligned} \check{\mathbf{r}}_0^T (\mathbf{p}_{i+1} - \beta_i \mathbf{p}_i) &= \check{\mathbf{r}}_0^T (\mathbf{r}_{i+1} - \beta_i \omega_i A\mathbf{p}_i) = & (25) \\ &= \check{\mathbf{r}}_0^T \mathbf{r}_{i+1} - \left( \frac{\check{\mathbf{r}}_0^T \mathbf{r}_{i+1}}{\check{\mathbf{r}}_0^T A\mathbf{p}_i} \right) \check{\mathbf{r}}_0^T A\mathbf{p}_i = 0 & (26) \end{aligned}$$

□

Notice that we can express the recurrences of BiCGStab in the following convenient form

$$\mathbf{s}_i = \mathbf{r}_i - \alpha_i A\mathbf{p}_i \tag{27}$$

$$\mathbf{r}_{i+1} = (I - \omega_i A)\mathbf{s}_i \tag{28}$$

$$\mathbf{q}_i = (I - \omega_i A)\mathbf{p}_i \tag{29}$$

$$\mathbf{p}_{i+1} = \mathbf{r}_{i+1} + \beta_i \mathbf{q}_i \tag{30}$$

**Corollary 3.** *Let auxiliary vector $\boldsymbol{y}_i = \boldsymbol{s}_i + \beta_i \boldsymbol{p}_i$, then directions $\boldsymbol{p}_{i+1} = (I - \omega_i A)\boldsymbol{y}_i$ and*

$$\check{\boldsymbol{r}}_0^T A\boldsymbol{y}_i = 0 \tag{31}$$

*as long as $\omega_i \neq 0$.*

*Proof.* Notice that using recurrences (28), (29) and (30) we have

$$\mathbf{p}_{i+1} = \mathbf{r}_{i+1} + \beta_i \mathbf{q}_i = (I - \omega_i A)\mathbf{s}_i + \beta_i (I - \omega_i A)\mathbf{p}_i = (I - \omega_i A)(\mathbf{s}_i + \beta_i \mathbf{p}_i) \tag{32}$$

and using first and last relationships in Theorem 2 we have

$$\begin{aligned} \check{\mathbf{r}}_0^T A(\mathbf{s}_i + \beta_i \mathbf{p}_i) &= -\frac{1}{\omega_i} \check{\mathbf{r}}_0^T A(-\omega_i \mathbf{s}_i - \beta_i \omega_i \mathbf{p}_i) = -\frac{1}{\omega_i} \check{\mathbf{r}}_0^T (\mathbf{s}_i - \omega_i A\mathbf{s}_i - \beta_i \omega_i A\mathbf{p}_i) \\ &= -\frac{1}{\omega_i} \check{\mathbf{r}}_0^T (\mathbf{r}_{i+1} - \beta_i \omega_i A\mathbf{p}_i) = -\frac{1}{\omega_i} \check{\mathbf{r}}_0^T (\mathbf{p}_{i+1} - \beta_i \mathbf{p}_i) = 0 \end{aligned} \tag{33}$$

□

These properties are interesting by themselves, but it is difficult to rely only on them to setup a linear system for finding scalars $\alpha$, $\omega$ and $\beta$ similarly to CG and BiCG methods because they involve a single vector corresponding to the shadow residual $\check{\mathbf{r}}_0$, rather than all directions $\mathbf{p}$ in CG or shadow directions $\check{\mathbf{p}}$ in BiCG. Therefore, we have to find additional conditions and a different way of setting up the s-step method.

Let us now assume that we have $s$ directions $\mathbf{p}_i$ and $\mathbf{s}_i$ available at once, then we have

$$\begin{aligned} \mathbf{x}_{i+s} &= \mathbf{x}_i + \alpha_i \mathbf{p}_i + ... + \alpha_{i+s-1}\mathbf{p}_{i+s-1} + \omega_i \mathbf{s}_i + ... + \omega_{i+s-1}\mathbf{s}_{i+s-1} \\ &= \mathbf{x}_i + P_i \mathbf{a}_i + S_i \mathbf{o}_i \end{aligned} \tag{34}$$

where $P_i = [\mathbf{p}_i, ..., \mathbf{p}_{i+s-1}]$ and $S_i = [\mathbf{s}_i, ..., \mathbf{s}_{i+s-1}] \in \mathbb{R}^{n \times s}$ are tall matrices and $\mathbf{a}_i = [\alpha_i, ..., \alpha_{i+s-1}]^T$ and $\mathbf{o}_i = [\omega_i, ..., \omega_{i+s-1}]^T \in \mathbb{R}^s$ are small vectors. Consequently, residual can be expressed as

$$\mathbf{r}_{i+s} = \mathbf{f} - A\mathbf{x}_{i+s} = \mathbf{r}_i - A(P_i \mathbf{a}_i + S_i \mathbf{o}_i) \tag{35}$$

Also, let us assume that we have the monomial basis for Krylov subspace for $s$ iterations

$$R_i = [\mathbf{r}_i, A\mathbf{r}_i, ..., A^{2(s-1)}\mathbf{r}_i] \tag{36}$$

and monomial basis for directions

$$P_i = [\mathbf{p}_i, A\mathbf{p}_i, ..., A^{2(s-1)}\mathbf{p}_i] \tag{37}$$

Notice that using recurrences (27) - (30) we can write the following

$$[\mathbf{s}_i, A\mathbf{s}_i, \mathbf{q}_i, A\mathbf{q}_i] = \left[\mathbf{r}_i, A\mathbf{r}_i, A^2\mathbf{r}_i, \mathbf{p}_i, A\mathbf{p}_i, A^2\mathbf{p}_i\right] C \tag{38}$$

$$\left[\mathbf{r}_{i+1}, \mathbf{p}_{i+1}\right] = [\mathbf{s}_i, A\mathbf{s}_i, \mathbf{q}_i, A\mathbf{q}_i]\, D \tag{39}$$

where transitional matrices of scalar coefficients $C$ and $D$ are

$$\begin{aligned} C = C^{(1)}C^{(2)} &= \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 0 & & \\ & & 1 & 0 & \\ -\alpha_i & & & 1 & 0 \\ & -\alpha_i & & & 1 \end{pmatrix} \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -\omega_i & 1 \\ & & & -\omega_i \end{pmatrix} \\ &= \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 0 & \\ & & 1 & 0 \\ -\alpha_i & & -\omega_i & 1 \\ & -\alpha_i & & -\omega_i \end{pmatrix} \end{aligned} \tag{40}$$

and

$$D = D^{(1)}D^{(2)} = \begin{pmatrix} 1 & & \\ -\omega_i & 0 & \\ & 1 & 0 \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ & \beta_i \\ & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ -\omega_i & -\omega_i \\ & \beta_i \\ & 0 \end{pmatrix} \tag{41}$$

Also, note that for $B = CD$ and $k \geq 0$ we have

$$A^k[\mathbf{r}_{i+1}, \mathbf{p}_{i+1}] = A^k[\mathbf{r}_i, A\mathbf{r}_i, A^2\mathbf{r}_i, \mathbf{p}_i, A\mathbf{p}_i, A^2\mathbf{p}_i]B \tag{42}$$

Therefore, letting $B_i \in \mathbb{R}^{4s-2 \times 4s-6}$ follow the same pattern as $B$ we have

$$[\mathbf{r}_{i+1}, ..., A^{2(s-2)}\mathbf{r}_{i+1}, \mathbf{p}_{i+1}, ..., A^{2(s-2)}\mathbf{p}_{i+1}] = [\mathbf{r}_i, ..., A^{2(s-1)}\mathbf{r}_i, \mathbf{p}_i, ..., A^{2(s-1)}\mathbf{p}_i]B_i \tag{43}$$

and consequently

$$[\mathbf{r}_{i+s-1}, \mathbf{p}_{i+s-1}] = [\mathbf{r}_i, ..., A^{2(s-1)}\mathbf{r}_i, \mathbf{p}_i, ..., A^{2(s-1)}\mathbf{p}_i]B_i...B_{i+s-2} \tag{44}$$

Hence, we can use (44) to build directions $\mathbf{s}_i$, $\mathbf{p}_i$ and residuals $\mathbf{r}_i$ after $s-1$ iterations, as long as we are able to compute the scalars $\alpha$, $\omega$ and $\beta$ used in transitional matrices.

Let us now compute the matrix $W \in \mathbb{R}^{4s-2 \times 4s-2}$ and vector $\mathbf{w} \in \mathbb{R}^{4s-2}$ such that

$$W_i = [R_i, P_i]^T[R_i, P_i] \tag{45}$$

$$\mathbf{w}_i^T = \check{\mathbf{r}}_0^T[R_i, P_i] = [\mathbf{g}_i^T, \mathbf{h}_i^T] \tag{46}$$

and let $W_i(j,k)$ denote $j$-th row and $k$-th column element of matrix $W_i$, while $\mathbf{w}_i(j)$ denotes the $j$-th element of vector $\mathbf{w}_i$. Also, let us use 1-based indexing.

Then,

$$\alpha_i = \frac{\check{\mathbf{r}}_0^T\mathbf{r}_i}{\check{\mathbf{r}}_0^TA\mathbf{p}_i} = \frac{\mathbf{g}_i(1)}{\mathbf{h}_i(2)} \tag{47}$$

Further, after updating

$$W_i^{'} = {C_i^{(1)}}^T W_i C_i^{(1)} \tag{48}$$

we have

$$\omega_i = \frac{\mathbf{s}_i^TA\mathbf{s}_i}{(A\mathbf{s}_i)^T(A\mathbf{s}_i)} = \frac{W_i^{'}(1,2)}{W_i^{'}(2,2)} \tag{49}$$

and, after updating

$$\mathbf{w}_i^{'} = (C_iD_i^{(1)})^T\mathbf{w}_i \tag{50}$$

we obtain

$$\beta_i = \left(\frac{\check{\mathbf{r}}_0^T\mathbf{r}_{i+1}}{\check{\mathbf{r}}_0^T\mathbf{r}_i}\right)\left(\frac{\alpha_i}{\omega_i}\right) = \left(\frac{\mathbf{w}_i^{'}(1)}{\mathbf{g}_i(1)}\right)\left(\frac{\alpha_i}{\omega_i}\right) \tag{51}$$

Finally,

$$W_{i+1} = (C_i^{(2)}D_i)^TW_i^{'}(C_i^{(2)}D_i) = B_i^TW_iB_i \tag{52}$$

and

$$\mathbf{w}_{i+1} = {D_i^{(2)}}^T\mathbf{w}_i^{'} = B_i^T\mathbf{w}_i \tag{53}$$

Finally, putting all the equations (34) - (53) together we obtain the pseudo-code for s-step BiCGStab iterative method, shown in Alg. 5.

**Algorithm 5** S-Step BiCGStab

Let $A \in \mathbb{R}^{n \times n}$ be a nonsingular coefficient matrix and let $\mathbf{x}_0$ be an initial guess.
Compute $\mathbf{r}_0 = \mathbf{f} - A\mathbf{x}_0$ and set $\mathbf{p}_0 = \mathbf{r}_0$
Let $\check{\mathbf{r}}_0$ be arbitrary (for example $\check{\mathbf{r}}_0 = \mathbf{r}_0$) and let index $k = is + j$ (initially $k = 0$)
**for** $i = 0, 1, ...$until convergence **do**
    Compute $T = [[\mathbf{r}_k, \mathbf{p}_k], A[\mathbf{r}_k, \mathbf{p}_k], ..., A^{2s}[\mathbf{r}_k, \mathbf{p}_k]]$ ▷ Matrix-power kernel
    Let $R_i = [\mathbf{r}_k, A\mathbf{r}_k, \ ..., A^{2s}\mathbf{r}_k] = [\tilde{R}_i, A^{2s}\mathbf{r}_k]$ ▷ Extract $R$ from $T$
    Let $P_i = [\mathbf{p}_k, A\mathbf{p}_k, ..., A^{2s}\mathbf{p}_k] = [\tilde{P}_i, A^{2s}\mathbf{p}_k] = [\mathbf{p}_k, \bar{P}_i]$ ▷ Extract $P$ from $T$
    Compute $W_i = [R_i, P_i]^T[R_i, P_i]$ ▷ Block dot-products
    Compute $\mathbf{w}_i^T = \check{\mathbf{r}}_0^T[R_i, P_i] = [\mathbf{g}_i^T, \mathbf{h}_i^T]$
    Let $\tilde{.}$ and $\bar{.}$ indicate all but last and first elements of ., as shown for $R_i$ and $P_i$
    **for** $j = 0, ..., s-1$ **do**
        Set $\alpha_k = \frac{\mathbf{g}_k(1)}{\mathbf{h}_k(2)}$ and store $\mathbf{a}_i(j) = \alpha_k$ ▷ Note $\alpha_k = \frac{\check{\mathbf{r}}_0^T \mathbf{r}_k}{\check{\mathbf{r}}_0^T A\mathbf{p}_k}$
        Compute $S_k = \tilde{R}_k - \alpha_k \bar{P}_k$ ▷ $S_k = [\mathbf{s}_k, A\mathbf{s}_k, ..., A^{2(s-j)-1}\mathbf{s}_k]$
        Update $W_k^{'} = {C_k^{(1)}}^T W_k C_k^{(1)}$ ▷ Using $\mathbf{s}_k = \mathbf{r}_k - \alpha_k A\mathbf{p}_k$
        **if** $\sqrt{W_k^{'}(1,1)}/||\mathbf{r}_0||_2 \leq$ tol **then** break **end** ▷ Check $||\mathbf{s}_k||_2^2$ for early exit
        Set $\omega_k = \frac{W_k^{'}(1,2)}{W_k^{'}(2,2)}$ and store $\mathbf{o}_i(j) = \omega_k$ ▷ Note $\omega_k = \frac{\mathbf{s}_k^T A\mathbf{s}_k}{(A\mathbf{s}_k)^T(A\mathbf{s}_k)}$
        Compute $Q_k = \tilde{P}_k - \omega_k \bar{P}_k$ ▷ $Q_k = [\mathbf{q}_k, A\mathbf{q}_k, ..., A^{2(s-j)-1}\mathbf{q}_k]$
        **if** $(j == s-1)$ **then** break; ▷ Check for last iteration
        Compute $R_{k+1} = \tilde{S}_k - \omega_k \bar{S}_k$ ▷ $R_{k+1} = [\mathbf{r}_{k+1}, .., A^{2(s-j)-2}\mathbf{r}_{k+1}]$
        Update $\mathbf{w}_k^{'} = (C_k D_k^{(1)})^T \mathbf{w}_k$ ▷ Using $\mathbf{r}_{k+1} = \mathbf{s}_k - \omega_k A\mathbf{s}_k$
        Set $\beta_k = \left(\frac{\mathbf{w}_k^{'}(1)}{\mathbf{g}_k(1)}\right)\left(\frac{\alpha_k}{\omega_k}\right)$ ▷ Note $\beta_k = \left(\frac{\check{\mathbf{r}}_0^T \mathbf{r}_{k+1}}{\check{\mathbf{r}}_0^T \mathbf{r}_k}\right)\left(\frac{\alpha_k}{\omega_k}\right)$
        Compute $P_{k+1} = R_{k+1} + \beta_k \tilde{Q}_k$ ▷ $P_{k+1} = [\mathbf{p}_{k+1}, .., A^{2(s-j)-2}\mathbf{p}_{k+1}]$
        Update $W_{k+1} = B_k^T W_k B_k$ ▷ Using $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{q}_k$
        Update $\mathbf{w}_{k+1} = B_k^T \mathbf{w}_k$
    **end for**
    Let $P_i = [\mathbf{p}_i, ..., \mathbf{p}_{k-1}]$ and $S_i = [\mathbf{s}_i, ..., \mathbf{s}_{k-1}]$ have the constructed directions
    Compute $\mathbf{x}_{i+s} = \mathbf{x}_i + P_i\mathbf{a}_i + S_i\mathbf{o}_i$ ▷ Do not use last $\mathbf{s}_{k-1}$ if early exit
    Compute $\mathbf{r}_{i+s} = \mathbf{f} - A\mathbf{x}_{i+s}$ ▷ Note $k = is + s - 1$ unless early exit
    **if** $||\mathbf{r}_{i+s}||_2/||\mathbf{r}_0||_2 \leq$ tol **or** early exit **then** stop **end** ▷ Check Convergence
    Compute $\beta_k = \left(\frac{\check{\mathbf{r}}_0^T \mathbf{r}_{i+s}}{\mathbf{g}_k(1)}\right)\left(\frac{\mathbf{a}_i(s-1)}{\mathbf{o}_i(s-1)}\right)$ ▷ Note $\alpha_k = \mathbf{a}_i(s-1)$ and $\omega_k = \mathbf{o}_i(s-1)$
    Compute $\mathbf{p}_{i+s} = \mathbf{r}_{i+s} + \beta_k \mathbf{q}_k$
**end for**

It is worthwhile to mention again that updates using transitional matrices of scalar coefficients $B_i = C_i D_i$ can be expressed in terms of recurrences. In fact notice that matrices $C^{(1)}$, $C^{(2)}$, $D^{(1)}$, $D^{(2)}$ in equations (40) - (41) have one-to-one correspondence to recurrences in equations (27) - (30). We use a mix of transitional matrices and recurrences for clarity in the pseudo-code. However, we point out that latter allows for more efficient implementation of the algorithm in practice.

Also, notice that at each inner iteration the size of tall matrices $R_i$ and $P_i$ as well as small square matrix $W_i$ and vector $\mathbf{w}_i$ is reduced by 4 elements (columns for $R_i$ and $P_i$, rows/columns for $W_i$ and vector elements for $\mathbf{w}_i$). These 4 elements correspond to two extra powers of $A$ and $A^2$ applied to residual $\mathbf{r}$ and directions $\mathbf{p}$. Consequently, the updates can be done in-place preserving all previously computed residuals and directions without requiring extra memory storage for them.

Let us now introduce preconditioning into the s-step BiCGStab in a manner similar to the standard algorithm. Let the preconditioned linear system be

$$(AM^{-1})(M\mathbf{x}) = \mathbf{f} \tag{54}$$

Then, we can apply the s-step on the preconditioned system (54) and re-factor recurrences in terms of preconditioner $M = LU$. If we express everything in terms of "preconditioned" directions $M^{-1}P_i$ and $M^{-1}S_i$ and work with the original solution vector $\mathbf{x}_i$ we obtain the preconditioned s-step BiCGStab method, shown in Alg. 6.

Notice that tall matrices $R_i$, $P_i$, $V_i$ and $Z_i$ on lines $7-10$ can be constructed during the computation of $T$ on line 6 in Alg. 6. The key observation is that the computation of $T$ proceeds in the following fashion

$$\begin{aligned} [\mathbf{r}_i, \mathbf{p}_i], \quad & M^{-1}[\mathbf{r}_i, \mathbf{p}_i], \\ (AM^{-1})[\mathbf{r}_i, \mathbf{p}_i], \quad & M^{-1}(AM^{-1})[\mathbf{r}_i, \mathbf{p}_i], \\ & \ldots \\ (AM^{-1})^{2s-1}[\mathbf{r}_i, \mathbf{p}_i], & M^{-1}(AM^{-1})^{2s-1}[\mathbf{r}_i, \mathbf{p}_i], \\ (AM^{-1})^{2s}[\mathbf{r}_i, \mathbf{p}_i] & \end{aligned} \tag{55}$$

and therefore terms in first and second column define tall matrices $R_i$, $P_i$, $V_i$ and $Z_i$.

It is important to point out that in preconditioned s-step BiCGStab we carry both $R_i$, $P_i$ and $V_i$, $Z_i$ through inner $s-1$ iterations. The latter pair is needed to compute original $\mathbf{x}_{i+s}$ on line 30 and then residual $\mathbf{r}_{i+s}$ on line 31, while the former pair is needed to compute direction $\mathbf{p}_k$ on line 35 in Alg. 6. Both residual $\mathbf{r}_k$ and direction $\mathbf{p}_k$ are needed to compute $T$ in the next outer iteration. Notice that this is different from preconditioned s-step CG, where only preconditioned residual $\mathbf{r}_k$ is needed for the next outer iteration.

Finally, notice that preconditioned s-step BiCGStab performs an extra matrix-vector multiplication with $A$ and preconditioner solve with $M^{-1}$ per 1 iteration, when compared with $s$ iterations of the standard preconditioned BiCGStab method. The extra work happens during re-computation of residual at the end of every s-step iteration.

**Algorithm 6** Preconditioned S-Step BiCGStab

Let $A \in \mathbb{R}^{n\times n}$ be a nonsingular coefficient matrix and let $\mathbf{x}_0$ be an initial guess.
Let $M = LU$ be the preconditioner, so that $M^{-1} \approx A^{-1}$.
Compute $\mathbf{r}_0 = \mathbf{f} - A\mathbf{x}_0$ and set $\mathbf{p}_0 = \mathbf{r}_0$
Let $\check{\mathbf{r}}_0$ be arbitrary (for example $\check{\mathbf{r}}_0 = \mathbf{r}_0$) and let index $k = is + j$ (initially $k = 0$)
**for** $i = 0, 1, ...$until convergence **do**
    Compute $T = [[\mathbf{r}_k, \mathbf{p}_k], \hat{A}[\mathbf{r}_k, \mathbf{p}_k], ..., \hat{A}^{2s}[\mathbf{r}_k, \mathbf{p}_k]]$ with $\hat{A} = AM^{-1}$ ▷ Matrix-power kernel
    Let $R_i = [\mathbf{r}_k, \hat{A}\mathbf{r}_k, \ ..., \hat{A}^{2s}\mathbf{r}_k] = [\tilde{R}_i, \hat{A}^{2s}\mathbf{r}_k]$ ▷ Build $R$, $P$, $V$ and $Z$
    Let $P_i = [\mathbf{p}_k, \hat{A}\mathbf{p}_k, ..., \hat{A}^{2s}\mathbf{p}_k] = [\tilde{P}_i, \hat{A}^{2s}\mathbf{p}_k] = [\mathbf{p}_k, \bar{P}_i]$ ▷ during comput. of $T$
    Let $V_i = M^{-1}[\mathbf{r}_k, \hat{A}\mathbf{r}_k, \ ..., \hat{A}^{2s-1}\mathbf{r}_k] = M^{-1}\tilde{R}_i$
    Let $Z_i = M^{-1}[\mathbf{p}_k, \hat{A}\mathbf{p}_k, ..., \hat{A}^{2s-1}\mathbf{p}_k] = M^{-1}\tilde{P}_i$
    Compute $W_i = [R_i, P_i]^T[R_i, P_i]$ ▷ Block dot-products
    Compute $\mathbf{w}_i^T = \quad \check{\mathbf{r}}_0^T[V_i, Z_i] = [\mathbf{g}_i^T, \mathbf{h}_i^T]$
    Let $\tilde{.}$ and $\bar{.}$ indicate all but last and first elements of ., as shown for $R_i$ and $P_i$
    **for** $j = 0, ..., s-1$ **do**
        Set $\alpha_k = \frac{\mathbf{g}_k(1)}{\mathbf{h}_k(2)}$ and store $\mathbf{a}_i(j) = \alpha_k$ ▷ Note $\alpha_k = \frac{\check{\mathbf{r}}_0^T\mathbf{r}_k}{\check{\mathbf{r}}_0^T A\mathbf{p}_k}$
        Compute $[S_k, S_k'] = [\tilde{R}_k, \tilde{V}_k] - \alpha_k[\bar{P}_k, \bar{Z}_k]$ ▷ $S_k = \ [\mathbf{s}_k, A\mathbf{s}_k, ..., A^{2(s-j)-1}\mathbf{s}_k]$
        Update $W_k' = {C_k^{(1)}}^T W_k C_k^{(1)}$ ▷ Using $\mathbf{s}_k = \mathbf{r}_k - \alpha_k A\mathbf{p}_k$
        **if** $\sqrt{W_k'(1,1)}/||\mathbf{r}_0||_2 \leq$ tol **then** break **end** ▷ Check $||\mathbf{s}_k||_2^2$ for early exit
        Set $\omega_k = \frac{W_k'(1,2)}{W_k'(2,2)}$ and store $\mathbf{o}_i(j) = \omega_k$ ▷ Note $\omega_k = \frac{\mathbf{s}_k^T A\mathbf{s}_k}{(A\mathbf{s}_k)^T(A\mathbf{s}_k)}$
        Compute $[Q_k, Q_k'] = [\tilde{P}_k, \tilde{Z}_k] - \omega_k[\bar{P}_k, \bar{Z}_k]$ ▷ $Q_k = [\mathbf{q}_k, A\mathbf{q}_k, ..., A^{2(s-j)-1}\mathbf{q}_k]$
        **if** $(j == s-1)$ **then** break; ▷ Check for last iteration
        Compute $[R_{k+1}, V_{k+1}] = [\tilde{S}_k, \tilde{S}_k'] - \omega_k[\bar{S}_k, \bar{S}_k']$ ▷ $R_{k+1} = [\mathbf{r}_{k+1}, .., A^{2(s-j)-2}\mathbf{r}_{k+1}]$
        Update $\mathbf{w}_k' = (C_k D_k^{(1)})^T \mathbf{w}_k$ ▷ Using $\mathbf{r}_{k+1} = \mathbf{s}_k - \omega_k A\mathbf{s}_k$
        Set $\beta_k = \left(\frac{\mathbf{w}_k'(1)}{\mathbf{g}_k(1)}\right)\left(\frac{\alpha_k}{\omega_k}\right)$ ▷ Note $\beta_k = \left(\frac{\check{\mathbf{r}}_0^T\mathbf{r}_{k+1}}{\check{\mathbf{r}}_0^T\mathbf{r}_k}\right)\left(\frac{\alpha_k}{\omega_k}\right)$
        Compute $[P_{k+1}, Z_{k+1}] = [R_{k+1}, V_{k+1}] + \beta_k[\tilde{Q}_k, \tilde{Q}_k']$ ▷ $P_{k+1} = [\mathbf{p}_{k+1}, .., A^{2(s-j)-2}\mathbf{p}_{k+1}]$
        Update $W_{k+1} = B_k^T W_k B_k$ ▷ Using $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k\mathbf{q}_k$
        Update $\mathbf{w}_{k+1} = B_k^T \mathbf{w}_k$
    **end for**
    Let $Z_i = M^{-1}[\mathbf{p}_i, ..., \mathbf{p}_{k-1}]$ and $S_i' = M^{-1}[\mathbf{s}_i, ..., \mathbf{s}_{k-1}]$ have the constructed directions
    Compute $\mathbf{x}_{i+s} = \mathbf{x}_i + Z_i\mathbf{a}_i + S_i'\mathbf{o}_i$ ▷ Do not use last $\mathbf{s}_{k-1}$ if early exit
    Compute $\mathbf{r}_{i+s} = \mathbf{f} - A\mathbf{x}_{i+s}$ ▷ Note $k = is + s - 1$ unless early exit
    **if** $||\mathbf{r}_{i+s}||_2/||\mathbf{r}_0||_2 \leq$ tol **or** early exit **then** stop **end** ▷ Check Convergence
    Compute $\mathbf{r}_{i+s}' = M^{-1}\mathbf{r}_{i+s}$
    Compute $\beta_k = \left(\frac{\check{\mathbf{r}}_0^T\mathbf{r}_{i+s}'}{\mathbf{g}_k(1)}\right)\left(\frac{\mathbf{a}_i(s-1)}{\mathbf{o}_i(s-1)}\right)$ ▷ Note $\alpha_k = \mathbf{a}_i(s-1)$ and $\omega_k = \mathbf{o}_i(s-1)$
    Compute $\hat{\mathbf{p}}_{i+s} = \hat{\mathbf{r}}_{i+s} + \beta_k\mathbf{q}_k$
**end for**

# 4 Numerical Experiments

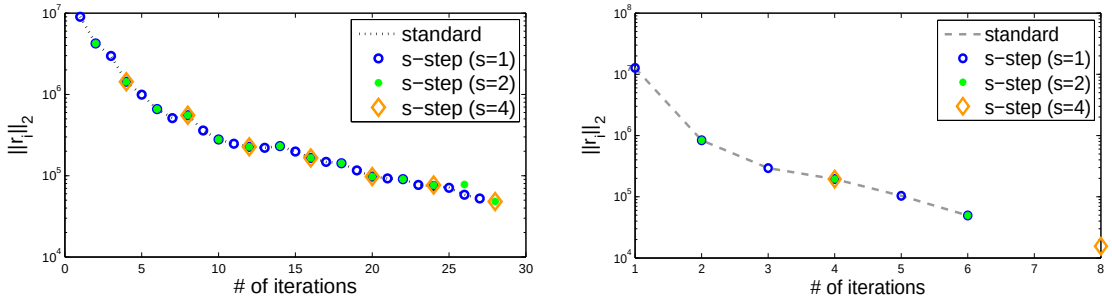
In this section we study the numerical behavior of the preconditioned s-step CG and BiCGStab iterative methods. In order to facilitate comparisons, we use the same matrices as previous investigations of their standard and block counterparts [13, 14]. The seven SPD and five nonsymmetric matrices from UFSMC [19] with the respective number of rows (m), columns (n=m) and non-zero elements (nnz) are grouped and shown according to their increasing order in Tab. 1.

We compare the s-step iterative methods using a reference MATLAB implementation. We let the initial guess be zero and the RHS $\mathbf{f} = A\mathbf{e}$ where $\mathbf{e} = (1, \ldots, 1)^T$. Also, unless stated otherwise we let the stopping criteria be the maximum number of iterations 40 or relative residual $||\mathbf{r}_i||_2/||\mathbf{r}_0||_2 < 10^{-2}$, where $\mathbf{r}_i = \mathbf{f} - A\mathbf{x}_i$ is the residual at $i$-th iteration. These modest stopping criteria allow us to illustrate the behavior of the algorithms without being significantly affected by the accumulation of floating point errors through iterations, which is addressed later on separately with different stopping criteria that match previous studies [13, 14].

| # | Matrix | m,n | nnz | SPD | Application |
|---|---|---|---|---|---|
| 1. | offshore | 259,789 | 4,242,673 | yes | Geophysics |
| 2. | af_shell3 | 504,855 | 17,562,051 | yes | Mechanics |
| 3. | parabolic_fem | 525,825 | 3,674,625 | yes | General |
| 4. | apache2 | 715,176 | 4,817,870 | yes | Mechanics |
| 5. | ecology2 | 999,999 | 4,995,991 | yes | Biology |
| 6. | thermal2 | 1,228,045 | 8,580,313 | yes | Thermal Simulation |
| 7. | G3_circuit | 1,585,478 | 7,660,826 | yes | Circuit Simulation |
| 8. | FEM_3D_thermal2 | 147,900 | 3,489,300 | no | Mechanics |
| 9. | thermomech_dK | 204,316 | 2,846,228 | no | Mechanics |
| 10. | ASIC_320ks | 321,671 | 1,316,085 | no | Circuit Simulation |
| 11. | cage13 | 445,315 | 7,479,343 | no | Biology |
| 12. | atmosmodd | 1,270,432 | 8,814,880 | no | Atmospheric Model. |

Table 1: Symmetric positive definite (SPD) and nonsymmetric test matrices

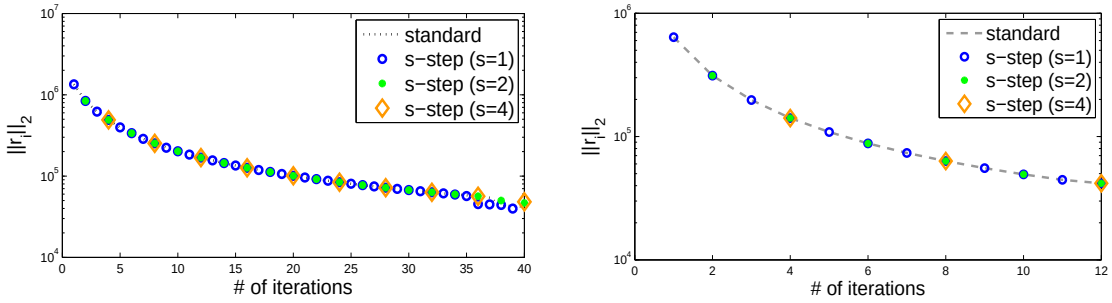
First, let us illustrate that s-step iterative methods indeed produce the same approximation to the solution $\mathbf{x}_i$ after $i$ iterations as their standard counterparts after $i \times s$ iterations. Notice that when we plot $||\mathbf{r}_i||_2$ through iterations for standard scheme in a grey line and for s-step scheme with $s = 1$ in blue circle, $s = 2$ in green star and $s = 4$ in orange diamond, these markings lie on the same curve and coincide in strides of $s$ iterations, as can be seen in Fig. 1 and 2. The only difference might happen in the last iteration where standard schemes can stop at an arbitrary iteration number, while s-step schemes must proceed to the next multiple of $s$. This indicates that both un- and preconditioned s-step CG and BiCGStab indeed produce the same solution $\mathbf{x}_i$ as standard iterative methods.




(a) Conjugate Gradient (CG)

(b) Preconditioned CG

Figure 1: Plot of convergence of standard and s-step CG for $s = 1, 2, 4$ on af_shell3 matrix



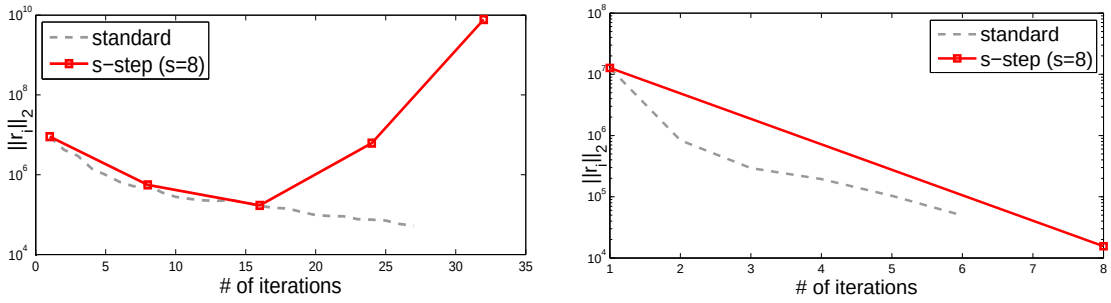

(a) BiConjugate Gradient Stabilized (BiCGStab)

(b) Preconditioned BiCGStab

Figure 2: Plot of convergence of standard and s-step BiCGStab for $s = 1, 2, 4$ on atmosmmodd matrix

Second, let us make an important observation about numerical stability of s-step methods. As we have mentioned earlier, the vectors in the monomial basis of the Krylov subspace used in s-step methods can become linearly dependent relatively quickly. In fact, if we plot $||\mathbf{r}_i||_2$ through iterations for $s = 8$ in red square, we can easily identify several occasions where these markings do not coincide with the grey line used for the standard scheme, see Fig. 3 and 4. Notice that in the case of s-step CG on Fig. 3 (a) the s-step method diverged, while in the case of s-step BiCGStab on Fig 4 (a) the method exited early, because intermediate value $||s||_2$ fell below a specified threshold. This behavior is in line with the observations made in [6] that for $s > 5$ s-step CG may suffer from loss of orthogonality and that preconditioned s-step CG would likely fair better than its unpreconditioned version.

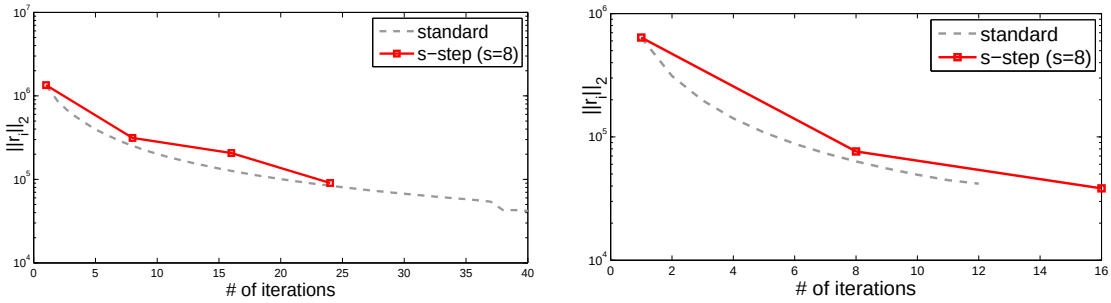
Third, let us look at the convergence of the preconditioned s-step iterative method for different stopping criteria. In particular, we are interested in what happens when we ask the algorithm to achieve a tighter tolerance $||\mathbf{r}_i||_2/||\mathbf{r}_0||_2 < 10^{-7}$ and allow a larger maximum number of iterations. Notice that the residuals computed by the s-step methods always start following the ones computed by standard algorithms, see Fig 5 and 6. In some cases, the s-step methods match their standard counterparts exactly until the solution is found, as shown for offshore matrix on Fig. 5 (a). In other cases, they follow the residuals approximately, while still finding the solution towards the end, as shown for atmosmodd




(a) Conjugate Gradient (CG)

(b) Preconditioned CG

Figure 3: Plot of convergence of standard and s-step CG for $s = 8$ on af_shell3 matrix




(a) BiConjugate Gradient Stabilized (BiCGStab)

(b) Preconditioned BiCGStab

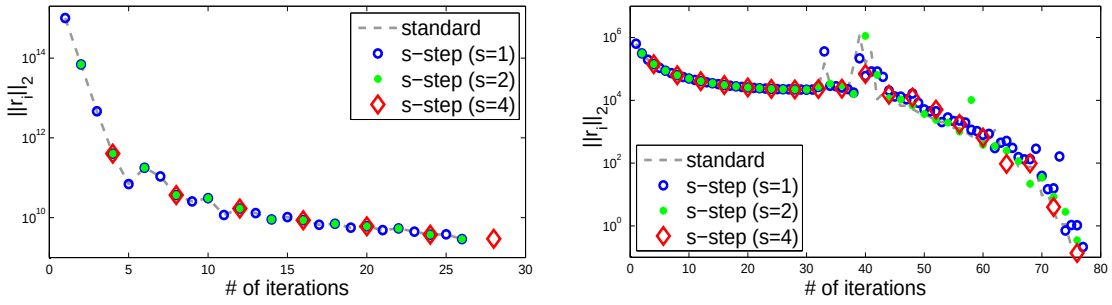
Figure 4: Plot of convergence of standard and s-step BiCGStab for $s = 8$ on atmosmmodd matrix

matrix on Fig. 5 (b). However, there are also cases where the residuals computed by standard and s-step methods initially coincide, but the latter diverge from the solution towards the end, as shown for matrices af_shell3 and apache2 on Fig. 6.

These empirical results lead us to conclude that s-step methods are often better suited for linear systems, where we are looking for an approximate solution with a relatively relaxed tolerance, such as $10^{-2}$, and expect to find it in modest number of iterations, for example $< 80$ iterations. Also, we recall that $s \leq 5$ is recommended for most cases.

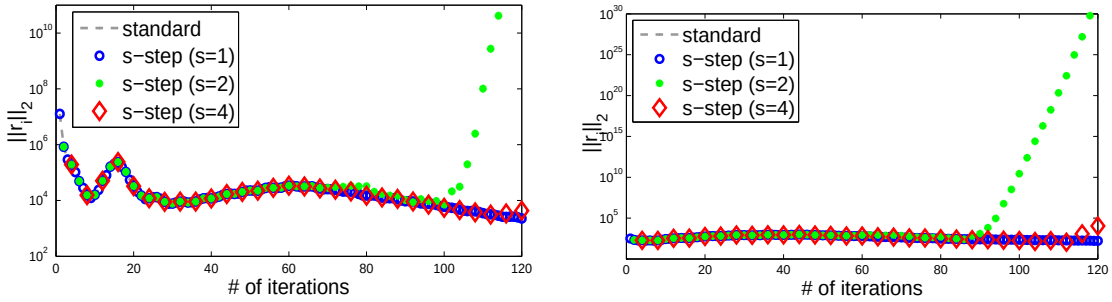
Finally, reverting to using our original stopping criteria that were stated on page 14, we plot $-\log \frac{||\mathbf{r}_i||_2}{||\mathbf{r}_0||_2}$ obtained by the preconditioned standard and s-step methods for all the matrices on Fig. 7. Notice that because we plot the negative of the log, the higher is the value on the plot in Fig. 7, the lower is the relative residual.

Notice that for most matrices the value of the relative residual across all schemes is either the same or slightly better for s-step schemes, because the latter stop at iteration number that is multiple of $s$, which might be slightly higher than the iteration at which the standard algorithm has stopped. There are also a few cases, such as thermomech_dK corresponding to matrix 9, where the relative residual for s-step method with $s = 8$ is worse, due to loss of orthogonality of the vectors in the monomial basis.




(a) offshore (b) atmosmodd

Figure 5: Plot of convergence of preconditioned methods for tighter tolerance $||\mathbf{r}_i||_2/||\mathbf{r}_0||_2 < 10^{-7}$




(a) af_shell3 matrix (b) apache2 matrix

Figure 6: Plot of convergence of preconditioned methods for tighter tolerance $||\mathbf{r}_i||_2/||\mathbf{r}_0||_2 < 10^{-7}$

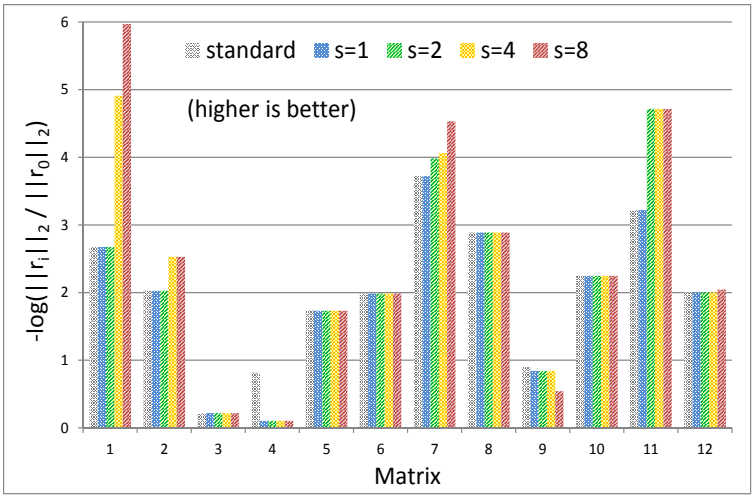



Figure 7: Plot of $-\log \frac{||\mathbf{r}_i||_2}{||\mathbf{r}_0||_2}$ obtained by preconditioned standard and s-step methods

The detailed results of the numerical experiments are shown in Tab. 2 and 3.

| | Standard | | s-step (s=1) | | s-step (s=2) | | s-step (s=4) | | s-step (s=8) | |
|---|---|---|---|---|---|---|---|---|---|---|
| # | # it. | $\frac{\|\mathbf{r}_i\|_2}{\|\mathbf{r}_0\|_2}$ | # it. | $\frac{\|\mathbf{r}_i\|_2}{\|\mathbf{r}_0\|_2}$ | # it. | $\frac{\|\mathbf{r}_i\|_2}{\|\mathbf{r}_0\|_2}$ | # it. | $\frac{\|\mathbf{r}_i\|_2}{\|\mathbf{r}_0\|_2}$ | # it. | $\frac{\|\mathbf{r}_i\|_2}{\|\mathbf{r}_0\|_2}$ |
| 1. | 15 | 9.18E-03 | 15 | 9.18E-03 | 8 | 8.35E-03 | 4 | 8.35E-03 | 2 | 8.35E-03 |
| 2. | 27 | 9.97E-03 | 27 | 9.96E-03 | 14 | 9.14E-03 | 7 | 9.13E-03 | 5 | 6.38E+06 |
| 3. | 40 | 4.03E+00 | 40 | 4.03E+00 | 20 | 4.03E+00 | 10 | 4.03E+00 | 5 | 4.03E+00 |
| 4. | 40 | 1.97E-01 | 40 | 1.97E-01 | 20 | 1.20E+06 | 10 | 1.51E+12 | 5 | 8.54E+04 |
| 5. | 40 | 1.38E-01 | 40 | 1.38E-01 | 20 | 1.38E-01 | 10 | 1.38E-01 | 5 | 1.38E-01 |
| 6. | 40 | 2.46E-02 | 40 | 2.46E-02 | 20 | 2.46E-02 | 10 | 2.46E-02 | 5 | 2.46E-02 |
| 7. | 1 | 1.70E-03 | 1 | 1.70E-03 | 1 | 1.57E-03 | 1 | 1.37E-03 | 1 | 7.54E-04 |
| 8. | 25.5 | 9.70E-03 | 26 | 9.70E-03 | 13 | 9.70E-03 | 7 | 9.70E-03 | 1 | 1.09E-01 |
| 9. | 40 | 2.21E-01 | 40 | 1.79E-01 | 20 | 1.96E-01 | 1 | 1.78E-01 | 5 | NaN |
| 10. | 40 | 1.16E-02 | 40 | 3.47E-02 | 20 | 1.91E-02 | 1 | 1.69E-01 | 1 | 1.69E-01 |
| 11. | 1.5 | 2.67E-03 | 2 | 2.67E-03 | 1 | 2.67E-03 | 1 | 2.67E-03 | 1 | 2.67E-03 |
| 12. | 39.5 | 9.89E-03 | 39 | 9.41E-03 | 20 | 1.10E-02 | 10 | 1.14E-02 | 3 | 2.14E-02 |

Table 2: Results for standard and s-step unpreconditioned CG and BiCGStab methods

| | Standard | | s-step (s=1) | | s-step (s=2) | | s-step (s=4) | | s-step (s=8) | |
|---|---|---|---|---|---|---|---|---|---|---|
| # | # it. | $\frac{\|\mathbf{r}_i\|_2}{\|\mathbf{r}_0\|_2}$ | # it. | $\frac{\|\mathbf{r}_i\|_2}{\|\mathbf{r}_0\|_2}$ | # it. | $\frac{\|\mathbf{r}_i\|_2}{\|\mathbf{r}_0\|_2}$ | # it. | $\frac{\|\mathbf{r}_i\|_2}{\|\mathbf{r}_0\|_2}$ | # it. | $\frac{\|\mathbf{r}_i\|_2}{\|\mathbf{r}_0\|_2}$ |
| 1. | 2 | 2.12E-03 | 2 | 2.12E-03 | 1 | 2.12E-03 | 1 | 1.23E-05 | 1 | 1.07E-06 |
| 2. | 6 | 9.38E-03 | 6 | 9.38E-03 | 3 | 9.38E-03 | 2 | 2.95E-03 | 1 | 2.95E-03 |
| 3. | 40 | 6.04E-01 | 40 | 6.04E-01 | 20 | 6.04E-01 | 10 | 6.04E-01 | 5 | 6.04E-01 |
| 4. | 40 | 1.51E-01 | 40 | 7.85E-01 | 20 | 7.85E-01 | 10 | 7.85E-01 | 5 | 7.85E-01 |
| 5. | 40 | 1.83E-02 | 40 | 1.85E-02 | 20 | 1.85E-02 | 10 | 1.85E-02 | 5 | 1.85E-02 |
| 6. | 40 | 1.03E-02 | 40 | 1.03E-02 | 20 | 1.03E-02 | 10 | 1.03E-02 | 5 | 1.03E-02 |
| 7. | 1 | 1.90E-04 | 1 | 1.90E-04 | 1 | 1.03E-04 | 1 | 8.58E-05 | 1 | 2.93E-05 |
| 8. | 1.5 | 1.29E-03 | 2 | 1.29E-03 | 1 | 1.29E-03 | 1 | 1.29E-03 | 1 | 1.29E-03 |
| 9. | 30 | 1.26E-01 | 40 | 1.44E-01 | 20 | 1.44E-01 | 10 | 1.44E-01 | 1 | 2.86E-01 |
| 10. | 0.5 | 5.63E-03 | 1 | 5.63E-03 | 1 | 5.63E-03 | 1 | 5.63E-03 | 1 | 5.63E-03 |
| 11. | 1 | 6.01E-04 | 1 | 6.01E-04 | 1 | 1.92E-05 | 1 | 1.92E-05 | 1 | 1.92E-05 |
| 12. | 11.5 | 9.87E-03 | 12 | 9.87E-03 | 6 | 9.87E-03 | 3 | 9.87E-03 | 2 | 9.04E-03 |

Table 3: Results for standard and s-step preconditioned CG and BiCGStab methods

## 5 Conclusion

In this paper we made an overview of the existing s-step CG iterative method and developed a novel s-step BiCGStab iterative method. We have also introduced preconditioning to both s-step algorithms, maintaining the extra work performed at each $s$ iterations to the extra matrix-vector multiplication and a preconditioning solve.

We have discussed advantages of these methods, such as the use of matrix-power kernel and block dot-products, as well as their disadvantages, such as the loss of orthogonality of the monomial Krylov subspace basis and extra work performed per iteration. Also, we have pointed out the connection between s-step and communication-avoiding algorithms.

Finally, we performed numerical experiments to validate the theory behind the s-step methods, and will look at their performance in the future. These algorithms can potentially be a better alternative to their standard counterparts when modest tolerance and number of iterations are required to solve a linear system on parallel platforms. Also, they could be used for a larger class of problems if the numerical stability issues related to the monomial basis are resolved.

## 6 Acknowledgements

The author would like to acknowledge Erik Boman and Michael Garland for their useful comments and suggestions.

## References

[1] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, PA, 1994.

[2] E. Carson, N. Knight and J. Demmel, *Avoiding Communication in Nonsymmetric Lanczos-Based Krylov Subspace Methods*, SIAM J. Sci. Comput., Vol. 35, pp. 42-61, 2012.

[3] E. Carson, *Communication-Avoiding Krylov Subspace Methods in Theory and Practice*, Ph.D. Thesis, University of California - Berkeley, 2015.

[4] A. T. Chronopoulos *A Class of Parallel Iterative Methods on Multiprocessor*, Ph.D. Thesis, University of Illinois - Urbana-Champagne, 1987.

[5] A. T. Chronopoulos, *S-Step Iterative Methods for (Non)Symmetric (In)Definite Linear Systems*, SIAM Journal on Numerical Analysis, Vol. 28, pp. 1776-1789, 1991.

[6] A. T. Chronopoulos and C. W. Gear, *S-Step Iterative Methods for Symmetric Linear Systems*, Journal of Computational and Applied Mathematics, Vol. 25, pp. 153-168, 1989.

[7] A. T. Chronopoulos and C. D. Swanson, *Parallel Iterative S-step Methods for Unsymmetric Linear Systems*, Parallel Computing. Vol. 22, pp. 623-641, 1996.

[8] A. T. Chronopoulos and S. K. Kim, *S-Step Orthomin and GMRES Implemented on Parallel Computers*, Technical Report UMSI 90/43R, University of Minnesota Supercomputing Institute, 1990.

[9] A. T. Chronopoulos and S. K. Kim, *S-step Lanczos and Arnoldi Methods on Parallel Computers*, Technical Report UMSI 90/14R, University of Minnesota Supercomputing Institute, Minneapolis, 1990.

[10] M. M. Dehnavi, J. Demmel, D. Giannacopoulos, Y. El-Kurdi, *Communication-Avoiding Krylov Techniques on Graphics Processing Units*, IEEE Trans. on Magnetics, Vol. 49, pp. 1749-1752, 2013.

[11] A. El Guennouni, K. Jbilou and H. Sadok, *A Block Version of BiCGStab for Linear Systems with Multiple Right-Hand-Sides*, ETNA, **16**, pp. 129–142 (2003).

[12] M. Hoemmen, *Communication-Avoiding Krylov Subspace Methods*, Ph.D. Thesis, University of California - Berkeley, 2010.

[13] M. Naumov, *Parallel Incomplete-LU and Cholesky Factorization in the Preconditioned Iterative Methods on the GPU*, Nvidia Technical Report, 3, 2012.

[14] M. Naumov, *Preconditioned Block-Iterative Methods on GPUs*, Proc. International Association of Applied Mathematics and Mechanics, PAMM, Vol. 12, pp. 11-14, 2012.

[15] D. P. O'Leary, *The Block Conjugate Gradient Algorithm and Related Methods*, Linear Algebra Appl., Vol. 29, pp. 293-322, 1980.

[16] J. van Rosendale, *Minimizing Inner Product Data Dependence in Conjugate Gradient Iteration*, Technical Report 172178, NASA Langley Research Center, 1983.

[17] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, 2nd Ed., 2003.

[18] I. Yamazaki, S. Rajamanickam, E. G. Boman, M. Hoemmen, M. A. Heroux and S. Tomov, *Domain Decomposition Preconditioners for Communication-Avoiding Krylov Methods on Hybrid CPU/GPU Cluster*, Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 933-944, 2014.

[19] The University of Florida Sparse Matrix Collection (UFSMC), http://www.cise.ufl.edu/research/sparse/matrices/