A subspace-conjugate gradient method for Kronecker-structured equations

Martina Iannacito

joint work with Davide Palitta and Valeria Simoncini

HAW visiting

January 28, 2026

Finanziato dall'Unione europea NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Multiterm matrix equation – I

Consider the **multiterm Sylvester equation**

$$\mathbf{A}_1\mathbf{X}\mathbf{B}_1 + \cdots + \mathbf{A}_m\mathbf{X}\mathbf{B}_m = \mathbf{C}$$

where $\mathbf{A}_k \in \mathbb{R}^{n_A \times n_A}$ and $\mathbf{B}_k \in \mathbb{R}^{n_B \times n_B}$ are symmetric, and $\mathbf{C}$ is low-rank (s_C).

Multiterm matrix equation – I

Consider the **multiterm Sylvester equation**

$$\mathbf{A}_1\mathbf{X}\mathbf{B}_1 + \cdots + \mathbf{A}_m\mathbf{X}\mathbf{B}_m = \mathbf{C}$$

where $\mathbf{A}_k \in \mathbb{R}^{n_A \times n_A}$ and $\mathbf{B}_k \in \mathbb{R}^{n_B \times n_B}$ are symmetric, and $\mathbf{C}$ is low-rank (s_C).

They arise in different fields, e.g.,

- statistics and probability
- control and system theory
- signal processing
- uncertanty quantification
- physical phenomena simulation

Multiterm matrix equation in control and system theory

Consider the continuous-time linear system

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}_1\mathbf{u} \\ \mathbf{y} &= \mathbf{B}_2^*\mathbf{x} \end{aligned} \tag{1}$$

where $\mathbf{x}$ is the *model state*, $\mathbf{u}$ is the *input*, $\mathbf{y}$ is the *output*, $\mathbf{A}$, $\mathbf{B}_1$ and $\mathbf{B}_2$ are invariant. If $\mathbf{A}$ is stable[1], then the solutions $\mathbf{P}$ and $\mathbf{Q}$ of the Lyapunov equations

$$\begin{aligned} \mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^* + \mathbf{B}_1\mathbf{B}_1^* &= 0 \\ \mathbf{A}^*\mathbf{Q} + \mathbf{Q}\mathbf{A} + \mathbf{B}_2\mathbf{B}_2^* &= 0 \end{aligned}$$

are called *controllability* and *observability* Gramians, respectively. They can be used to measure the energy transfers in the system (1).
See [Antoulas 2005; Simoncini 2016] for further details.

Solving multiterm matrix equations

If the number of terms $m = 2$

- if n_A and n_B are small, direct methods can be used, e.g., Bartels–Stewart method
- otherwise, need iterative methods, e.g., projection methods

Solving multiterm matrix equations

If the number of terms $m = 2$

- if n_A and n_B are small, direct methods can be used, e.g., Bartels–Stewart method
- otherwise, need iterative methods, e.g., projection methods

If the number of terms $m > 2$

- direct methods can be used only through the Kronecker form, that is

$$(\mathbf{B}_1^\top \otimes_{\mathrm{K}} \mathbf{A}_1 + \cdots + \mathbf{B}_m^\top \otimes_{\mathrm{K}} \mathbf{A}_m)\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{C}).$$

  prohibitive for moderate n_A, n_B.

Solving multiterm matrix equations

If the number of terms $m = 2$

- if n_A and n_B are small, direct methods can be used, e.g., Bartels–Stewart method
- otherwise, need iterative methods, e.g., projection methods

If the number of terms $m > 2$

- direct methods can be used only through the Kronecker form, that is

$$(\mathbf{B}_1^\top \otimes_{\mathrm{K}} \mathbf{A}_1 + \cdots + \mathbf{B}_m^\top \otimes_{\mathrm{K}} \mathbf{A}_m)\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{C}).$$

  prohibitive for moderate n_A, n_B.
- iterative methods, such as
  - matrix-oriented Krylov methods
  - projection methods
  - fixed-point iterations
  - Riemannian optimization schemes

Matrix-oriented CG – I

Let $\mathcal{L}(\mathbf{X}) = \mathbf{A}_1\mathbf{X}\mathbf{B}_1 + \cdots + \mathbf{A}_m\mathbf{X}\mathbf{B}_m$ for any $\mathbf{X} \in \mathbb{R}^{n\times n}$, with $n = n_A = n_B$, and $\mathbf{A}_h, \mathbf{B}_h$ are symmetric.

Matrix-oriented CG – I

Let $\mathcal{L}(\mathbf{X}) = \mathbf{A}_1\mathbf{X}\mathbf{B}_1 + \cdots + \mathbf{A}_m\mathbf{X}\mathbf{B}_m$ for any $\mathbf{X} \in \mathbb{R}^{n\times n}$, with $n = n_A = n_B$, and $\mathbf{A}_h, \mathbf{B}_h$ are symmetric.

Let $\left(\mathcal{L}(\mathbf{Y})\right)^\top = \mathcal{L}(\mathbf{Y})$ for $\mathbf{Y}$ symmetric, so that we can work with $\mathbf{Y} = YY^\top$ with $Y \in \mathbb{R}^{n\times r}$.

Matrix-oriented CG – I

Let $\mathcal{L}(\mathbf{X}) = \mathbf{A}_1\mathbf{X}\mathbf{B}_1 + \cdots + \mathbf{A}_m\mathbf{X}\mathbf{B}_m$ for any $\mathbf{X} \in \mathbb{R}^{n\times n}$, with $n = n_A = n_B$, and $\mathbf{A}_h, \mathbf{B}_h$ are symmetric.

Let $\left(\mathcal{L}(\mathbf{Y})\right)^\top = \mathcal{L}(\mathbf{Y})$ for $\mathbf{Y}$ symmetric, so that we can work with $\mathbf{Y} = YY^\top$ with $Y \in \mathbb{R}^{n\times r}$.

Given an initial guess $\mathbf{X}_0 \in \mathbb{R}^{n\times n}$, the matrix-oriented CG iterate is

$$\mathbf{X}_{k+1} = \mathbf{X}_k + \alpha_k \mathbf{P}_k$$

where $\alpha_k \in \mathbb{R}$.

The direction matrix iterate is

$$\mathbf{P}_{k+1} = \mathbf{R}_{k+1} + \beta_k \mathbf{P}_k$$

where $\mathbf{R}_{k+1} = \mathbf{C} - \mathcal{L}(\mathbf{X}_{k+1})$ is the residual and $\beta_k \in \mathbb{R}$.

Matrix-oriented CG – II

The dense matrices

- $\mathbf{X}_{k+1}$ for the *iterative solution*
- $\mathbf{R}_{k+1}$ for the *residual*
- $\mathbf{P}_{k+1}$ for the *directions*

have size $(n \times n)$ matrices with n potentially very large

If $\mathbf{C}$ is low-rank, $\mathbf{C} = CC^\top$ with C an $(n \times s)$ matrix, we can use the *factored forms*

$$\mathbf{X}_{k+1} = [X_k, \sqrt{\alpha_k} P_k][X_k, \sqrt{\alpha_k} P_k]^\top$$

where $\mathbf{X}_k = X_k X_k^\top$ and $\mathbf{P}_k = P_k P_k^\top$ with X_k and P_k thin matrices.

Limits of matCG

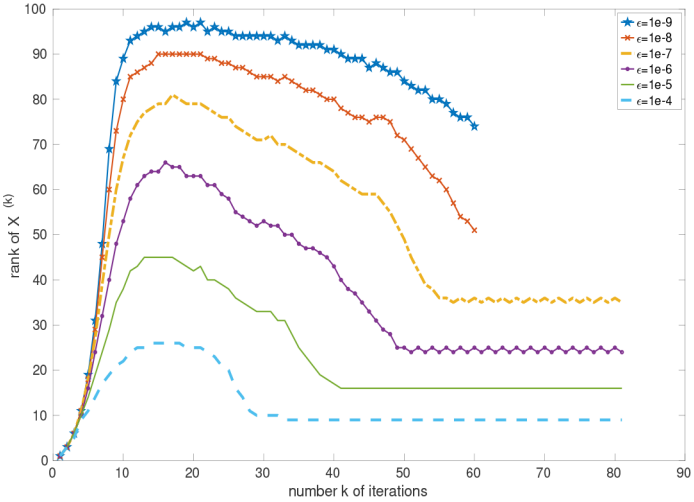
Usually the blocks X_{k+1}, R_{k+1} and P_{k+1} get larger accumulating redundant information

Introduce truncation on X_{k+1}, R_{k+1} and P_{k+1} at fixed rank or precision

While computational costs are controlled, side-effects are

- delayed or stagnating convergence;
- hard to control the rank.

Rank vs iteration [Simoncini and Hao 2023, Fig. 4]



Subspace CG

Idea: modifying α_k and β_k nature to improve convergence

Subspace CG

Idea: modifying α_k and β_k nature to improve convergence

Define $\Phi : \mathbb{R}^{n\times n} \to \mathbb{R}$ such that

$$\Phi(\mathbf{X}) = \frac{1}{2}\langle \mathcal{L}(\mathbf{X}), \mathbf{X}\rangle - \langle \mathbf{C}, \mathbf{X}\rangle.$$

Consider the minimization problem

$$\text{Find } \mathbf{X} \in \mathbb{R}^{n\times n} \quad \text{s.t.} \quad \mathbf{X} = \arg\min_{\mathbf{X}\in\mathbb{R}^{n\times n}} |\Phi(\mathbf{X})|.$$

and the new recurrence

$$\mathbf{X}_{k+1} = \mathbf{X}_k + P_k \boldsymbol{\alpha}_k P_k^\top$$

where $\boldsymbol{\alpha}_k \in \mathbb{R}^{s_k\times s_k}$ and $P_k \in \mathbb{R}^{n\times s_k}$.

Subspace CG – I

- recurrence for the direction

$$\mathbf{P}_{k+1} = \mathbf{R}_{k+1} + P_k \beta_k P_k^\top$$

where the residual remains $\mathbf{R}_{k+1} = \mathbf{C} - \mathcal{L}(\mathbf{X}_{k+1})$ and $\beta_k \in \mathbb{R}^{s_k \times s_k}$.

- the direction factor matrix $P_{k+1} \in \mathbb{R}^{n \times s_{k+1}}$ is computed from

$$\mathbf{P}_{k+1} = P_{k+1} \gamma_{k+1} P_{k+1}^\top$$

where $\gamma_{k+1} \in \mathbb{R}^{s_{k+1} \times s_{k+1}}$.

Why matrix coefficients (might) work better?

Let $\mathbf{x}_j^{(k+1)}$ denotes the j-th column of $\mathbf{X}_{k+1}$, the iterative solution at the $k+1$ iteration.

In the matrix-CG, $\mathbf{x}_j^{(k+1)} = \mathbf{x}_j^{(k)} + \mathbf{u}$ where $\mathbf{u}$ is the updated defined as

$$\mathbf{u} = \alpha_k (p_{j,1}\mathbf{p}_1 + \cdots + p_{j,s_k}\mathbf{p}_{s_k})$$

where $P_k = [\mathbf{p}_1, \ldots, \mathbf{p}_{s_k}]$ and $P_k(i,j) = p_{i,j}$.

Why matrix coefficients (might) work better?

Let $\mathbf{x}_j^{(k+1)}$ denotes the j-th column of $\mathbf{X}_{k+1}$, the iterative solution at the $k+1$ iteration.

In the matrix-CG, $\mathbf{x}_j^{(k+1)} = \mathbf{x}_j^{(k)} + \mathbf{u}$ where $\mathbf{u}$ is the updated defined as

$$\mathbf{u} = \alpha_k(p_{j,1}\mathbf{p}_1 + \cdots + p_{j,s_k}\mathbf{p}_{s_k})$$

where $P_k = [\mathbf{p}_1, \ldots, \mathbf{p}_{s_k}]$ and $P_k(i,j) = p_{i,j}$.

In the new method, the j-th column of $\mathbf{X}_{k+1}$ is computed as $\mathbf{x}_j^{(k+1)} = \mathbf{x}_j^{(k)} + \mathbf{u}$ where the updated $\mathbf{u}$ is

$$\Big(\sum_{i=1}^{s_k} \alpha_{1,i}\, p_{j,i}\Big)\mathbf{p}_1 + \cdots + \Big(\sum_{i=1}^{s_k} \alpha_{s_k,i}\, p_{j,i}\Big)\mathbf{p}_{s_k}$$

with $\boldsymbol{\alpha}_k(i,j) = \alpha_{i,j}$.

New matrix coefficients – I

- $\boldsymbol{\alpha}_k$ is the solution of

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^{s_k \times s_k}} \Phi(\mathbf{X}_k + P_k \boldsymbol{\alpha} P_k^\top)$$

that corresponds the orthogonality condition

$$\mathrm{vec}(\mathbf{R}_{k+1}) \perp \mathrm{range}(P_k \otimes_{\mathrm{K}} P_k).$$

New matrix coefficients – I

- $\boldsymbol{\alpha}_k$ is the solution of

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^{s_k \times s_k}} \Phi(\mathbf{X}_k + P_k \boldsymbol{\alpha} P_k^\top)$$

that corresponds the orthogonality condition

$$\mathrm{vec}(\mathbf{R}_{k+1}) \perp \mathrm{range}(P_k \otimes_{\mathrm{K}} P_k).$$

In the matrix-CG, $\alpha_k \in \mathbb{R}$ is computed to satisfy the orthogonality condition

$$\mathrm{vec}(\mathbf{R}_k - \alpha_k \mathcal{L}(\mathbf{P}_k)) \perp \mathrm{vec}(\mathbf{P}_k).$$

recalling that $\mathbf{R}_{k+1} = \mathbf{R}_k - \alpha_k \mathcal{L}(\mathbf{P}_k)$ and $\mathbf{P}_k = P_k P_k^\top$.

New matrix coefficients – I

- $\boldsymbol{\alpha}_k$ is the solution of

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^{s_k \times s_k}} \Phi(\mathbf{X}_k + P_k \boldsymbol{\alpha} P_k^\top)$$

that corresponds the orthogonality condition

$$\text{vec}(\mathbf{R}_{k+1}) \perp \text{range}(P_k \otimes_{\text{K}} P_k). \qquad \leftarrow \text{ orthogonality w.r.t. subspace of size } s_k^2 \text{ of } \mathbb{R}^{n^2}$$

In the matrix-CG, $\alpha_k \in \mathbb{R}$ is computed to satisfy the orthogonality condition

$$\text{vec}(\mathbf{R}_k - \alpha_k \mathcal{L}(\mathbf{P}_k)) \perp \text{vec}(\mathbf{P}_k).$$

recalling that $\mathbf{R}_{k+1} = \mathbf{R}_k - \alpha_k \mathcal{L}(\mathbf{P}_k)$ and $\mathbf{P}_k = P_k P_k^\top$.

New matrix coefficients – I

- $\boldsymbol{\alpha}_k$ is the solution of

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^{s_k \times s_k}} \Phi(\mathbf{X}_k + P_k \boldsymbol{\alpha} P_k^\top)$$

that corresponds the orthogonality condition

$$\text{vec}(\mathbf{R}_{k+1}) \perp \text{range}(P_k \otimes_{\text{K}} P_k). \qquad \leftarrow \text{ orthogonality w.r.t. subspace of size } s_k^2 \text{ of } \mathbb{R}^{n^2}$$

In the matrix-CG, $\alpha_k \in \mathbb{R}$ is computed to satisfy the orthogonality condition

$$\text{vec}(\mathbf{R}_k - \alpha_k \mathcal{L}(\mathbf{P}_k)) \perp \text{vec}(\mathbf{P}_k). \qquad \leftarrow \text{ orthogonality w.r.t. subspace of size } 1 \text{ of } \mathbb{R}^{n^2}$$

recalling that $\mathbf{R}_{k+1} = \mathbf{R}_k - \alpha_k \mathcal{L}(\mathbf{P}_k)$ and $\mathbf{P}_k = P_k P_k^\top$.

New matrix coefficients – II

- β_k is s.t. the updated directions, $\mathbf{P}_{k+1}$, are $\mathcal{L}$-orthogonal w.r.t. the previous ones,

$$\text{vec}(\mathcal{L}(\mathbf{P}_{k+1})) \perp \text{range}(P_k \otimes_{\text{K}} P_k)$$

New matrix coefficients – II

- β_k is s.t. the updated directions, $\mathbf{P}_{k+1}$, are $\mathcal{L}$-orthogonal w.r.t. the previous ones,

$$\text{vec}(\mathcal{L}(\mathbf{P}_{k+1})) \perp \text{range}(P_k \otimes_{\text{K}} P_k)$$

In the matrix-CG, $\beta_k \in \mathbb{R}$ is computed to satisfy the orthogonality condition

$$\text{vec}(\mathcal{L}(\mathbf{P}_{k+1})) \perp \text{vec}(\mathbf{P}_k).$$

New matrix coefficients – II

- β_k is s.t. the updated directions, $\mathbf{P}_{k+1}$, are $\mathcal{L}$-orthogonal w.r.t. the previous ones,

$$\text{vec}(\mathcal{L}(\mathbf{P}_{k+1})) \perp \text{range}(P_k \otimes_{\text{K}} P_k) \qquad \leftarrow \text{ orthogonality w.r.t. subspace of size } s_k^2 \text{ of } \mathbb{R}^{n^2}$$

In the matrix-CG, $\beta_k \in \mathbb{R}$ is computed to satisfy the orthogonality condition

$$\text{vec}(\mathcal{L}(\mathbf{P}_{k+1})) \perp \text{vec}(\mathbf{P}_k).$$

New matrix coefficients – II

- β_k is s.t. the updated directions, $\mathbf{P}_{k+1}$, are $\mathcal{L}$-orthogonal w.r.t. the previous ones,

$$\text{vec}(\mathcal{L}(\mathbf{P}_{k+1})) \perp \text{range}(P_k \otimes_{\text{K}} P_k) \qquad \leftarrow \text{ orthogonality w.r.t. subspace of size } s_k^2 \text{ of } \mathbb{R}^{n^2}$$

In the matrix-CG, $\beta_k \in \mathbb{R}$ is computed to satisfy the orthogonality condition

$$\text{vec}(\mathcal{L}(\mathbf{P}_{k+1})) \perp \text{vec}(\mathbf{P}_k). \qquad \leftarrow \text{ orthogonality w.r.t. subspace of size } 1 \text{ of } \mathbb{R}^{n^2}$$

Computation of the matrix coefficients - I

- $\boldsymbol{\alpha}_k$ is the unique solution of

$$P_k^\top \mathcal{L}(P_k \boldsymbol{\alpha} P_k^\top - \mathbf{R}_k) P_k = 0$$

Computation of the matrix coefficients - I

- $\boldsymbol{\alpha}_k$ is the unique solution of

$$P_k^\top \mathcal{L}(P_k \boldsymbol{\alpha} P_k^\top - \mathbf{R}_k) P_k = 0$$

or equivalently

$$P_k^\top \mathcal{L}(P_k \boldsymbol{\alpha} P_k^\top) P_k = P_k^\top \mathbf{R}_k P_k.$$

Computation of the matrix coefficients - I

- $\boldsymbol{\alpha}_k$ is the unique solution of

$$P_k^\top \mathcal{L}(P_k \boldsymbol{\alpha} P_k^\top - \mathbf{R}_k) P_k = 0$$

or equivalently

$$P_k^\top \mathcal{L}(P_k \boldsymbol{\alpha} P_k^\top) P_k = P_k^\top \mathbf{R}_k P_k.$$

- $\boldsymbol{\beta}_k$ is the unique solution of

$$P_k^\top \mathcal{L}(P_k \boldsymbol{\beta} P_k^\top + \mathbf{R}_{k+1}) P_k = 0$$

or equivalently

$$P_k^\top \mathcal{L}(P_k \boldsymbol{\beta} P_k^\top) P_k = -P_k^\top \mathcal{L}(\mathbf{R}_{k+1}) P_k.$$

Computation of the matrix coefficients - II

Both equations defining $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ are linear matrix equations of the **same type** as the original problem but of **smaller size** $(s_k \times s_k)$.

Let $\tilde{\mathbf{A}}_h^{(k)}$ and $\tilde{\mathbf{B}}_h^{(k)}$ be $(s_k \times s_k)$ matrices for $h = 1, \ldots, \ell$ defined as

$$\tilde{\mathbf{A}}_h^{(k)} = P_k^\top \mathbf{A}_h P_k \qquad \text{and} \qquad \tilde{\mathbf{B}}_h^{(k)} = P_k^\top \mathbf{B}_h P_k.$$

Then $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ are solutions of

$$\begin{aligned}
\tilde{\mathbf{A}}_1^{(k)} \boldsymbol{\alpha} \tilde{\mathbf{B}}_1^{(k)} + \cdots + \tilde{\mathbf{A}}_m^{(k)} \boldsymbol{\alpha} \tilde{\mathbf{B}}_m^{(k)} &= P_k^\top \mathbf{R}_k P_k \\
\tilde{\mathbf{A}}_1^{(k)} \boldsymbol{\beta} \tilde{\mathbf{B}}_1^{(k)} + \cdots + \tilde{\mathbf{A}}_m^{(k)} \boldsymbol{\beta} \tilde{\mathbf{B}}_m^{(k)} &= -P_k^\top \mathcal{L}(\mathbf{R}_{k+1}) P_k
\end{aligned}$$

Practical enhancements

The entire method works also if $\mathcal{L}(\mathbf{X}) \neq (\mathcal{L}(\mathbf{X}))^\top$, but we distinguish between *left* (ℓ) and right (r) factors, e.g.,

$$\mathbf{X}_{k+1} = \mathbf{X}_k + P_k^{(\ell)} \boldsymbol{\alpha}_k \left(P_k^{(r)}\right)^\top.$$

Practical enhancements

The entire method works also if $\mathcal{L}(\mathbf{X}) \neq (\mathcal{L}(\mathbf{X}))^\top$, but we distinguish between *left* (ℓ) and right (r) factors, e.g.,

$$\mathbf{X}_{k+1} = \mathbf{X}_k + P_k^{(\ell)} \boldsymbol{\alpha}_k (P_k^{(r)})^\top.$$

To make Ss-CG method efficient, we consider some enhancements

- block representation;
- truncation at a prescribed `maxrank` value;
- inexact coefficients;
- stopping criterion.

Inexact coefficients

- if `maxrank` is sufficiently small, $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ can be computed solving

$$\Big(\sum_{j=1}^{m} \tilde{\mathbf{B}}_j^\top \otimes_{\mathrm{K}} \tilde{\mathbf{A}}_j\Big)\mathrm{vec}(\boldsymbol{\alpha}) = \mathrm{vec}(P_k^\top \mathbf{R}_k P_k)$$

$$\Big(\sum_{j=1}^{m} \tilde{\mathbf{B}}_j^\top \otimes_{\mathrm{K}} \tilde{\mathbf{A}}_j\Big)\mathrm{vec}(\boldsymbol{\beta}) = -\mathrm{vec}(P_k^\top \mathcal{L}(\mathbf{R}_{k+1}) P_k)$$

Inexact coefficients

- if `maxrank` is sufficiently small, $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ can be computed solving

$$\Big(\sum_{j=1}^{m} \tilde{\mathbf{B}}_j^\top \otimes_{\mathrm{K}} \tilde{\mathbf{A}}_j\Big)\mathrm{vec}(\boldsymbol{\alpha}) = \mathrm{vec}(P_k^\top \mathbf{R}_k P_k)$$

$$\Big(\sum_{j=1}^{m} \tilde{\mathbf{B}}_j^\top \otimes_{\mathrm{K}} \tilde{\mathbf{A}}_j\Big)\mathrm{vec}(\boldsymbol{\beta}) = -\mathrm{vec}(P_k^\top \mathcal{L}(\mathbf{R}_{k+1}) P_k)$$

- otherwise, use an iterative solver at prescribed precision getting

$$\tilde{\boldsymbol{\alpha}}_k = \boldsymbol{\alpha}_k + \boldsymbol{\epsilon}_k \qquad \text{and} \qquad \tilde{\boldsymbol{\beta}}_k = \boldsymbol{\beta}_k + \boldsymbol{\psi}_k$$

Warning

The orthogonality properties previously seen are lost. Due to the use of truncation, inexactness of the coefficient matrices has not a strong effect.

Stopping criterion

Since the computation of the exact residual might be expensive, we test if the relative difference between consecutive iterates is smaller than the prescribed accuracy `tol`, that is

$$||\mathbf{X}_{k+1} - \mathbf{X}_k||_F \,/\, ||\mathbf{X}_{k+1}||_F \leq \texttt{tol}$$

The difference between two iterates can be efficiently computed using the block structure and the trace properties!

Warning

It might be sensitive to ill-conditioning of the problem, thus the true residual is computed explicitly at completion.

Memory requirements

- each iterate in block format requires at most

$$(n_A + n_B)\texttt{maxrank} + \texttt{maxrank}^2 \qquad \text{units of memory}$$

- matrix coefficients requires s_k^2 reaching at most $\texttt{maxrank}$ units of memory
- dynamic truncation of the residual requires

$$(m \cdot \texttt{maxrank} + s_C)(n_A + n_B) \qquad \text{units of memory}$$

while randomized truncation requires at least

$$\texttt{maxrankR}(n_A + n_B) \qquad \text{units of memory}$$

Memory costs are comparable to those of the Truncated CG

Numerical example competitors

The multiterm matrix equations of the following numerical examples are solved

- Ss-CG
  - deter(ministic)
  - rand(omized)
- TPCG: Truncated matrix-oriented Preconditioned CG [Kressner et al. 2011; Simoncini and Hao 2023]
- R-NLCG: Riemannian, nonlinear CG [Bioli et al. 2025]
- MultiRB: projection method for finite element discretizations of differential equations with stochastic input [Powell et al. 2017]

Numerical example I – [Benner et al. 2013; Shank et al. 2016]

We consider the discretization of a heat model problem in $(0,1)^2$, and resulting in the equation

$$\mathbf{AX} + \mathbf{XA} + \mathbf{MXM}^\top = \mathbf{C}$$

where $\mathbf{A}$ is the discretization of the 2D Laplace operator, and $\mathbf{M} = \mathbf{NN}^\top$ that allocates the Robin conditions $\bar{\boldsymbol{n}} \cdot \nabla(x) = \delta u(x-1)$ on one of the domain boundaries, while zero Dirichlet conditions are imposed on the rest of the boundary.

In our experiments we consider $\delta \in \{0.5, 0.9\}$. We precondition with $\mathcal{L}_0(\mathbf{X}) = \mathbf{AX} + \mathbf{XA}$ by running $t_{ADI} = 8$ LR-ADI iterations.

Numerical example I – results

Example	n_A	maxrank	SSS (iter/alloc/rank)	TPCG	SsCG determ.	SsCG rand'zed
$\delta = 0.5$	102400	20		61.35 (16)	– (100)	– (100)
		30		22.78 (4)	17.93 (3)	18.17 (3)
			6.15 (7/126/31)			
	250000	20		– (100)	– (100)	– (100)
		30		60.84 (4)	64.46 (4)	64.45 (4)
			18.35 (7/139/31)			
$\delta = 0.9$	102400	40		– (100)	– (100)	– (100)
		50		310.72 (26)	**58.11** (5)	58.52 (5)
		–	– (50/ /)			
	250000	50		2401.90 (93)	– (100)	– (100)
		60		936.39 (30)	**119.55** (4)	120.43 (4)
		–	– (50/ /)			

– no conv.

Table: Running time in seconds, and in parenthesis the number of iterations. Stopping tolerance $\texttt{tol} = 10^{-6}$. For SSS, number of iterations, the subspace total memory allocation for length n_A vectors and the solution rank are reported.

Numerical example II – [Bioli et al. 2025]

The stationary diffusion equation

$$-\nabla \cdot (\kappa \nabla u) = 0 \qquad \text{in} \qquad (0,1) \times (0,1)$$

with Dirichlet boundary conditions and semi-separable diffusion coefficient:

$$\kappa(x,y) = \sum_{j=0}^{m} \delta_j \kappa_{x,j}(x) \kappa_{y,j}(y) = 1 + \sum_{j=1}^{m_k - 1} \frac{10^j}{j!} x^j y^j.$$

The resulting multiterm linear equation is

$$\sum_{j=1}^{m_k} \delta_j \Big(\mathbf{A}_{j,x} \mathbf{X} \mathbf{D}_{j,y} + \mathbf{D}_{j,x} \mathbf{X} \mathbf{A}_{j,y} \Big) = \mathbf{C}$$

with $\mathbf{C}$ being rank 4, $m_k = 4$ and a total of 8 terms.

Numerical example II – results

n type	Precond	maxrank	R-NLCG	TPCG	SsCG determ.	SsCG rand.
10000	$\mathcal{P}_1$	20	– (100)	– (100)	– (100)	– (100)
	$\mathcal{P}_1$	40	– (100)	– (100)	1.08 (5)	**0.92** (5)
	$\mathcal{P}_1$	60	– (100)	– (100)	2.47 (5)	**2.34** (5)
	$\mathcal{P}_2$	20	**11.25 (36)**	11.42 (38)	– (100)	– (100)
	$\mathcal{P}_2$	40	*42.97 (36)	**15.54** (33)	– (100)	– (100)
	$\mathcal{P}_2$	60	*98.62 (35)	32.39 (28)	9.59 (5)	**8.37** (5)
102400	$\mathcal{P}_1$	20	– (100)	– (100)	– (100)	– (100)
	$\mathcal{P}_1$	40	†	– (100)	18.17 (6)	**8.74** (6)
	$\mathcal{P}_1$	60	†	– (100)	23.50 (5)	**16.93** (5)
	$\mathcal{P}_2$	20	**183.44** (41)	– (100)	– (100)	– (100)
	$\mathcal{P}_2$	40	†	446.94 (47)	– (100)	– (100)
	$\mathcal{P}_2$	60	†	884.20 (26)	115.73 (3)	**101.91** (3)

– no conv. * Lower final residual norm than other methods † Out of Memory

Table: Running time in seconds, and in parenthesis the number of iterations. Stopping tolerance $\texttt{tol} = 5 \cdot 10^{-6}$. Truncation tolerance $\texttt{tolrank} = 10^{-12}$. $\mathcal{P}_1$: one-term precond, $\mathcal{P}_2$: two-term precond, expensive.

Numerical example III – [Powell et al. 2017]

Let $\mathbf{A}_i$ and $\mathbf{B}_i$ be the data from [Powell et al. 2017, Example 5.1 and 5.2] coming from the discretization of a 2-dimensional elliptic PDE problem with correlated random inputs,

$$-\nabla \cdot (a(x,\omega)\nabla u) \quad = \quad f \quad \text{in } D, \quad u(x,\omega) = 0, \quad \text{on } \partial D.$$

with $\omega \in \Omega$ a sample space associated with a proper probability space, and $D \subset \mathbb{R}^2$ is the space domain, and

$$a(x,\omega) = \mu(x) + \sigma \sum_{j=1}^{\ell-1} \sqrt{\lambda_j}\phi_j(x)\xi_j(\omega),$$

where μ corresponds to the diffusion coefficient expected value, σ is the standard deviation, while (λ_j, ϕ_j) are the leading eigenpairs of the associated covariance matrix.

The right-hand side has rank 1, while the linear equation counts 10 terms.

Numerical example III – results

Example ($\ell = 10$)	n_A, n_B	maxrank	R-NLCG	MultiRB (spacedim/rank)	TPCG	Ss-CG determ.	Ss-CG rand'zed
[Ex.5.1]	16129, 2002	25	⋆5.58 (28)		6.55 (24)	– (100)	– (100)
		50	14.63 (26)		13.03 (16)	4.89 (8)	**3.20** (8)
		100	35.98 (25)		37.11 (16)	6.39 (6)	3.82 (6)
				6.89 (158/66)			
[Ex.5.2]	16129, 1287	60	– (100)		– (100)	– (100)	– (100)
		125	67.54 (38)		53.50 (18)	19.55 (12)	**11.83** (13)
		150	57.31 (24)		66.88 (14)	23.73 (11)	12.58 (11)
				12.76 (312/306)			

– no conv. ⋆ Final residual norm is *larger* than for other methods

Table: Running time in seconds, and in parenthesis the number of iterations. Stopping tolerance $\texttt{tol} = 10^{-6}$. Truncation tolerance $\texttt{tolrank} = 10^{-12}$. Best running times are in bold. For MultiRB the the final approximation space dimension and the final solution rank are reported.

Summary

We presented the Subspace CG, discussing

- matrix coefficients for the CG iterates
- new orthogonality conditions with richer subspaces
- computational enhancements based on randomization and inexactness
- memory costs comparable to TPCG
- promising on numerical examples

Further details on preconditioning, orthogonality and optimality can be found in

D. Palitta, M. Iannacito, and V. Simoncini. A Subspace–Conjugate Gradient Method for Linear Matrix Equations. *SIAM J. Matrix Anal. & Appl.*, 46(4):2197–2225, 2025.

References I

- Antoulas, A. C. (2005). *Approximation of Large-Scale Dynamical Systems*. Society for Industrial and Applied Mathematics.
- Benner, P. and T. Breiten (2013). "Low rank methods for a class of generalized Lyapunov equations and related issues". In: *Numer. Math.* 124, pp. 441–470.
- Bioli, I., D. Kressner, and L. Robol (2025). "Preconditioned Low-Rank Riemannian Optimization for Symmetric Positive Definite Linear Matrix Equations". In: *SIAM Journal on Scientific Computing* 47.2, A1091–A1116.
- Kressner, D. and C. Tobler (2011). "Low-Rank Tensor Krylov Subspace Methods for Parametrized Linear Systems". In: *SIAM. J. Matrix Anal. & Appl.* 32.4, pp. 1288–1316.
- Powell, C. E., D. Silvester, and V. Simoncini (2017). "An Efficient Reduced Basis Solver for Stochastic Galerkin Matrix Equations". In: *SIAM J. Sci. Comput.* 39.1, A141–A163.
- Shank, S. D., V. Simoncini, and D. B. Szyld (2016). "Efficient low-rank solutions of Generalized Lyapunov equations". In: *Numerische Mathematik* 134.2, pp. 327–342.

References II

- Simoncini, V. (2016). "Computational Methods for Linear Matrix Equations". In: *SIAM Review* 58.3, pp. 377–441.
- Simoncini, V. and Y. Hao (2023). "Analysis of the Truncated Conjugate Gradient Method for Linear Matrix Equations". In: *SIAM. J. Matrix Anal. & Appl.* 44.1, pp. 359–381.