A subspace-conjugate gradient method for linear matrix equations

Martina Iannacito

joint work with Davide Palitta and Valeria Simoncini

INdAM Workshop: Low-rank Structures and Numerical Methods in Matrix and Tensor Computations

September 3, 2025

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 - II

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

Multiterm matrix equation - II

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.

Multiterm matrix equation - II

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 $(\mathcal{L}(\mathbf{Y}))^\top = \mathcal{L}(\mathbf{Y})$ 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 $(\mathcal{L}(\mathbf{Y}))^\top = \mathcal{L}(\mathbf{Y})$ 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 $\mathbf{R}_{k+1}$ is the residual and $\beta_k \in \mathbb{R}$.

Matrix-oriented CG – II

$\mathbf{X}_k$, $\mathbf{P}_k$ and $\mathbf{R}_k$ are dense $(n \times n)$ matrices with n potentially very large

⇩

Relying on $\mathbf{C}$ 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

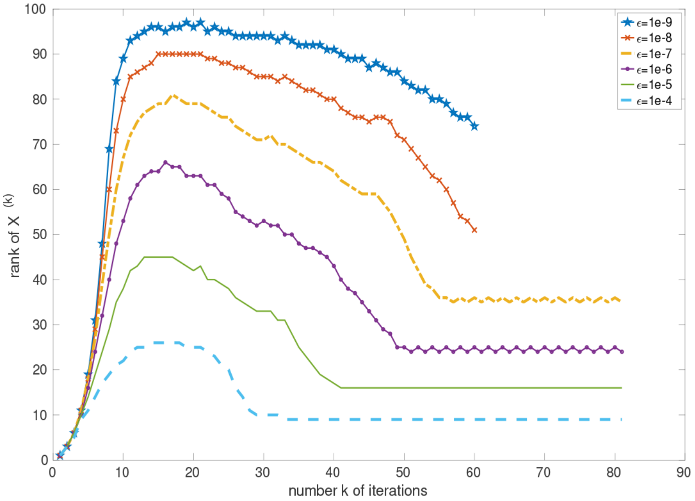
Usually the blocks get larger accumulating redundant information

Introduce truncation on X_k and P_k 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 et al. 2023, Fig. 4]



Subspace CG – I

Subspace CG

Idea: modifying α_k and β_k nature to improve convergence

Subspace CG – I

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 residual

$$\mathbf{R}_{k+1} = \mathbf{C} - \mathcal{L}(\mathbf{X}_{k+1}) = \mathbf{R}_k - \mathcal{L}(P_k \boldsymbol{\alpha}_k P_k^\top)$$

- recurrence for the direction

$$\mathbf{P}_{k+1} = \mathbf{R}_{k+1} + P_k \boldsymbol{\beta}_k P_k^\top$$

  where $\boldsymbol{\beta}_k \in \mathbb{R}^{s_k \times s_k}$

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

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

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

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

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

In the matCG, $\alpha_k \in \mathbb{R}$ is computed as

$$\alpha_k = \langle \mathbf{R}_k, \mathbf{P}_k\rangle / \langle \mathcal{L}(\mathbf{P}_k), \mathbf{P}_k\rangle = \text{tr}(\mathbf{R}_k^\top \mathbf{P}_k)/\text{tr}((\mathcal{L}(\mathbf{P}_k))^\top \mathbf{P}_k)$$

which corresponds to the orthogonality condition

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

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

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

In the matCG, $\alpha_k \in \mathbb{R}$ is computed as

$$\alpha_k = \langle \mathbf{R}_k, \mathbf{P}_k\rangle / \langle \mathcal{L}(\mathbf{P}_k), \mathbf{P}_k\rangle = \mathrm{tr}(\mathbf{R}_k^\top \mathbf{P}_k)/\mathrm{tr}((\mathcal{L}(\mathbf{P}_k))^\top \mathbf{P}_k)$$

which corresponds to the orthogonality condition

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

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

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

In the matCG, $\alpha_k \in \mathbb{R}$ is computed as

$$\alpha_k = \langle \mathbf{R}_k, \mathbf{P}_k\rangle / \langle \mathcal{L}(\mathbf{P}_k), \mathbf{P}_k\rangle = \text{tr}(\mathbf{R}_k^\top \mathbf{P}_k)/\text{tr}((\mathcal{L}(\mathbf{P}_k))^\top \mathbf{P}_k)$$

which corresponds to the orthogonality condition

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

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,

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

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,

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

In the matCG, $\beta_k \in \mathbb{R}$ is computed as

$$\beta_k = \mathrm{tr}(-\mathbf{R}_{k+1}^\top \mathcal{L}(\mathbf{P}_k))/\mathrm{tr}(\mathbf{P}_k^\top \mathcal{L}(\mathbf{P}_k))$$

which corresponds to the orthogonality condition

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

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,

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

  In the matCG, $\beta_k \in \mathbb{R}$ is computed as

$$\beta_k = \text{tr}(-\mathbf{R}_{k+1}^\top \mathcal{L}(\mathbf{P}_k))/\text{tr}(\mathbf{P}_k^\top \mathcal{L}(\mathbf{P}_k))$$

  which corresponds to 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 matCG, $\beta_k \in \mathbb{R}$ is computed as

  $$\beta_k = \text{tr}(-\mathbf{R}_{k+1}^\top \mathcal{L}(\mathbf{P}_k))/\text{tr}(\mathbf{P}_k^\top \mathcal{L}(\mathbf{P}_k))$$

  which corresponds to the orthogonality condition

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

New matrix coefficients – II

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

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

  In the matCG, $\beta_k \in \mathbb{R}$ is computed as

$$\beta_k = \mathrm{tr}(-\mathbf{R}_{k+1}^\top \mathcal{L}(\mathbf{P}_k))/\mathrm{tr}(\mathbf{P}_k^\top \mathcal{L}(\mathbf{P}_k))$$

  which corresponds to the orthogonality condition

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

The constraints on $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ guarantee that $\mathbf{P}_k$ is a descent direction, $\langle \nabla\Phi(\mathbf{X}_k), \mathbf{P}_k \rangle \leq 0$.

Computation of the matrix coefficients - I

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

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

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

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

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.,

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

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;
- inexact coefficients;
- stopping criterion.

Block representation I

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

$$\mathbf{X}_{k+1} = X_{k+1}^{(\ell)} \boldsymbol{\tau}_{k+1} (X_{k+1}^{(r)})^\top = [X_k^{(\ell)}, P_k^{(\ell)}] \begin{bmatrix} \boldsymbol{\tau}_k & \\ & \boldsymbol{\alpha}_k \end{bmatrix} [X_k^{(r)}, P_k^{(r)}]^\top$$

Block representation I

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

$$\mathbf{X}_{k+1} = X_{k+1}^{(\ell)} \boldsymbol{\tau}_{k+1} (X_{k+1}^{(r)})^\top = [X_k^{(\ell)}, P_k^{(\ell)}] \begin{bmatrix} \boldsymbol{\tau}_k & \\ & \boldsymbol{\alpha}_k \end{bmatrix} [X_k^{(r)}, P_k^{(r)}]^\top$$

- direction matrix $\mathbf{P}_{k+1} = \mathbf{R}_{k+1} + P_k^{(\ell)} \boldsymbol{\beta}_k (P_k^{(r)})^\top$ as

$$\mathbf{P}_{k+1} = P_{k+1}^{(\ell)} \boldsymbol{\gamma}_{k+1} (P_{k+1}^{(r)})^\top = [R_{k+1}^{(\ell)}, P_k^{(\ell)}] \begin{bmatrix} \boldsymbol{\rho}_k & \\ & \boldsymbol{\beta}_k \end{bmatrix} [R_{k+1}^{(r)}, P_k^{(r)}]^\top$$

Block representation II

- residual matrix $\mathbf{R}_{k+1} = \mathbf{C} - \mathcal{L}(\mathbf{X}_{k+1}) = \mathbf{C} - \mathcal{L}\left(X_{k+1}^{(\ell)} \boldsymbol{\tau}_{k+1} (X_{k+1}^{(r)})^\top\right)$ as

$$\mathbf{R}_{k+1} = R_{k+1}^{(\ell)} \boldsymbol{\rho}_{k+1} (R_{k+1}^{(r)})^\top = [C^{(\ell)}, \mathbf{A}_\star \bullet X_{k+1}^{(\ell)}]\, \boldsymbol{\rho}_{k+1}\, [C^{(r)}, \mathbf{B}_\star \bullet X_{k+1}^{(r)}]^\top$$

where

$$\boldsymbol{\rho}_{k+1} = \begin{bmatrix} \mathbb{I} & & & \\ & -\boldsymbol{\tau}_{k+1} & & \\ & & \ddots & \\ & & & -\boldsymbol{\tau}_{k+1} \end{bmatrix}, \qquad \begin{aligned} \mathbf{A}_\star \bullet Y &= [\mathbf{A}_1 Y, \ldots, \mathbf{A}_m Y] \\ \mathbf{B}_\star \bullet W &= [\mathbf{B}_1 W, \ldots, \mathbf{B}_m W] \end{aligned}$$

for $Y \in \mathbb{R}^{n_A \times p}$ and $W \in \mathbb{R}^{n_B \times p}$,

Block representation II

- residual matrix $\mathbf{R}_{k+1} = \mathbf{C} - \mathcal{L}(\mathbf{X}_{k+1}) = \mathbf{C} - \mathcal{L}\Big(X_{k+1}^{(\ell)} \boldsymbol{\tau}_{k+1} (X_{k+1}^{(r)})^\top\Big)$ as

$$\mathbf{R}_{k+1} = R_{k+1}^{(\ell)} \boldsymbol{\rho}_{k+1} (R_{k+1}^{(r)})^\top = [C^{(\ell)}, \mathbf{A}_\star \bullet X_{k+1}^{(\ell)}]\, \boldsymbol{\rho}_{k+1}\, [C^{(r)}, \mathbf{B}_\star \bullet X_{k+1}^{(r)}]^\top$$

where

$$\boldsymbol{\rho}_{k+1} = \begin{bmatrix} \mathbb{I} & & & \\ & -\boldsymbol{\tau}_{k+1} & & \\ & & \ddots & \\ & & & -\boldsymbol{\tau}_{k+1} \end{bmatrix}, \qquad \begin{aligned} \mathbf{A}_\star \bullet Y &= [\mathbf{A}_1 Y, \ldots, \mathbf{A}_m Y] \\ \mathbf{B}_\star \bullet W &= [\mathbf{B}_1 W, \ldots, \mathbf{B}_m W] \end{aligned}$$

for $Y \in \mathbb{R}^{n_A \times p}$ and $W \in \mathbb{R}^{n_B \times p}$,

Warning

For many terms, allocating $\mathbf{A}_\star \bullet X_{k+1}^{(\ell)}$ and $\mathbf{B}_\star \bullet X_{k+1}^{(r)}$ might not be possible!

.

Truncation – I

As the iterations progress, the rank grows and the number of columns in the blocks get larger

Truncation – I

As the iterations progress, the rank grows and the number of columns in the blocks get larger

⇩

Truncation strategies

Truncation – I

As the iterations progress, the rank grows and the number of columns in the blocks get larger

⇩

Truncation strategies

`QR` + `SVD` for solution and direction

1. $Q_h, S_h = \mathtt{QR}([X_k^{(h)}, P_k^{(h)}])$ for $h = \ell, r$;
2. $U_\ell, \Sigma, U_r = \mathtt{SVD}(S_\ell \boldsymbol{\tau}_{k+1} S_r^\top)$;
3. $X_{k+1}^{(h)} = Q_h U_h(:, 1 : j_k)$ for $h = \ell, r$;
4. $\boldsymbol{\tau}_{k+1} = \Sigma(1 : j_k, 1 : j_k)$

where j_k is computing considering a precision (`tolrank`) and a maximum allowed rank (`maxrank`).

Truncation – I

As the iterations progress, the rank grows and the number of columns in the blocks get larger

⇩

Truncation strategies

QR + SVD for solution and direction

1. $Q_h, S_h = \mathtt{QR}([X_k^{(h)}, P_k^{(h)}])$ for $h = \ell, r$;
2. $U_\ell, \Sigma, U_r = \mathtt{SVD}(S_\ell \boldsymbol{\tau}_{k+1} S_r^\top)$;
3. $X_{k+1}^{(h)} = Q_h U_h(:, 1:j_k)$ for $h = \ell, r$;
4. $\boldsymbol{\tau}_{k+1} = \Sigma(1:j_k, 1:j_k)$

where j_k is computing considering a precision (`tolrank`) and a maximum allowed rank (`maxrank`).

Ad hoc methods for residual

- dynamical
- randomized

Residual truncation: dynamical

Sequential combination of $2m$ QR and SVD to detect earlier linear dependencies

```
1 $Q_\ell, S_\ell = \mathtt{QR}([C_\ell, \mathbf{A}_1 X_k^{(\ell)} \boldsymbol{\tau}_k])$ and $Q_r, S_r = \mathtt{QR}([C_r, -\mathbf{B}_1 X_k^{(r)}])$
2 for $j = 2, \ldots, m$ do
3     $Q_\ell, S_{\ell,0} = \mathtt{QR}([Q_\ell, \mathbf{A}_j X_k^{(\ell)} \boldsymbol{\tau}_k])$ and $Q_r, S_{r,0} = \mathtt{QR}([Q_r, -\mathbf{B}_j X_k^{(r)}])$
4     $U_\ell, \Sigma_\ell, V_\ell = \mathtt{SVD}(S_{\ell,0})$ and $U_r, \Sigma_r, V_r = \mathtt{SVD}(S_{r,0})$
5     $Q_\ell = Q_\ell U_\ell(:, 1:j_k)$ and $Q_r = Q_r U_r(:, 1:j_k)$
6     $S_\ell = \left(\Sigma_\ell V_\ell^\top\right)(1:j_k, :) \begin{bmatrix} S_\ell & \\ & \mathbb{I} \end{bmatrix}$ and $S_r = \left(\Sigma_r V_r^\top\right)(1:j_k, :) \begin{bmatrix} S_r & \\ & \mathbb{I} \end{bmatrix}$
7 $R_k^{(\ell)} = Q_\ell$ and $R_k^{(r)} = Q_r$
8 $\boldsymbol{\rho}_k = S_\ell S_r^\top$
```

Residual truncation: randomized

Let G_ℓ and G_r Gaussian sketching matrices of size $(n_B \times \texttt{maxrankR})$ and $(n_A \times \texttt{maxrankR})$ respectively

1. compute $\mathbf{R}_k G_\ell$ (similarly $\mathbf{R}_k^\top G_r$) as

$$\mathbf{R}_k G_\ell = \Big(\mathbf{C} - \mathcal{L}(\mathbf{X}_k)\Big) G_\ell$$

Residual truncation: randomized

Let G_ℓ and G_r Gaussian sketching matrices of size $(n_B \times \texttt{maxrankR})$ and $(n_A \times \texttt{maxrankR})$ respectively

1. compute $\mathbf{R}_k G_\ell$ (similarly $\mathbf{R}_k^\top G_r$) as
$$\mathbf{R}_k G_\ell = \Big(\mathbf{C} - \mathcal{L}(\mathbf{X}_k)\Big) G_\ell = C^{(\ell)}\Big(\big(C^{(r)}\big)^\top G_\ell\Big) - \sum_{j=1}^{m} \mathbf{A}_j \Big(X_k^{(\ell)} \boldsymbol{\tau}_k \Big(\big(\mathbf{X}^{(r)}\big)^\top (\mathbf{B}_j G_\ell)\Big)\Big)$$
of size $(n_A \times \texttt{maxrankR})$ (similarly $(n_B \times \texttt{maxrankR})$)

Residual truncation: randomized

Let G_ℓ and G_r Gaussian sketching matrices of size $(n_B \times \texttt{maxrankR})$ and $(n_A \times \texttt{maxrankR})$ respectively

1. compute $\mathbf{R}_k G_\ell$ (similarly $\mathbf{R}_k^\top G_r$) as
$$\mathbf{R}_k G_\ell = \Big(\mathbf{C} - \mathcal{L}(\mathbf{X}_k)\Big) G_\ell = C^{(\ell)}\Big(\big(C^{(r)}\big)^\top G_\ell\Big) - \sum_{j=1}^{m} \mathbf{A}_j\Big(X_k^{(\ell)} \boldsymbol{\tau}_k\Big(\big(\mathbf{X}^{(r)}\big)^\top (\mathbf{B}_j G_\ell)\Big)\Big)$$
of size $(n_A \times \texttt{maxrankR})$ (similarly $(n_B \times \texttt{maxrankR})$)
2. $Q, \sim = \texttt{QR}(\mathbf{R}_k G_\ell)$ and $W, \sim = \texttt{QR}(\mathbf{R}_k^\top G_r)$, knowing that range$(Q) \approx$ range$(\mathbf{R}_k G_\ell)$ and range$(W) \approx$ range$(\mathbf{R}_k^\top G_r)$

Residual truncation: randomized

Let G_ℓ and G_r Gaussian sketching matrices of size $(n_B \times \texttt{maxrankR})$ and $(n_A \times \texttt{maxrankR})$ respectively

1. compute $\mathbf{R}_k G_\ell$ (similarly $\mathbf{R}_k^\top G_r$) as
$$\mathbf{R}_k G_\ell = \Big(\mathbf{C} - \mathcal{L}(\mathbf{X}_k)\Big) G_\ell = C^{(\ell)}\Big(\big(C^{(r)}\big)^\top G_\ell\Big) - \sum_{j=1}^{m} \mathbf{A}_j \Big(X_k^{(\ell)} \boldsymbol{\tau}_k \Big(\big(\mathbf{X}^{(r)}\big)^\top (\mathbf{B}_j G_\ell)\Big)\Big)$$
of size $(n_A \times \texttt{maxrankR})$ (similarly $(n_B \times \texttt{maxrankR})$)
2. $Q, \sim = \texttt{QR}(\mathbf{R}_k G_\ell)$ and $W, \sim = \texttt{QR}(\mathbf{R}_k^\top G_r)$, knowing that range$(Q) \approx$ range$(\mathbf{R}_k G_\ell)$ and range$(W) \approx$ range$(\mathbf{R}_k^\top G_r)$
3. compute $(\texttt{maxrankR} \times \texttt{maxrankR})$ matrix $Q^\top \mathbf{R}_k W$ as
$$Q^\top \mathbf{R}_k W = (Q^\top C^{(\ell)})(C^r)^\top W - \sum_{j=1}^{m} \Big(Q^\top \mathbf{A}_j X_k^{(\ell)}\Big) \boldsymbol{\tau}_k \Big(\big(\mathbf{X}^{(r)}\big)^\top \mathbf{B}_j W\Big)$$
4. $\hat{U}, \hat{\Sigma}, \hat{V} = \texttt{svd}(Q^\top \mathbf{R}_k W)$ truncating at $\texttt{maxrankR}$
5. $R_k^{(\ell)} = Q\hat{U}$, $R_k^{(r)} = W\hat{V}$ and $\boldsymbol{\rho}_k = \hat{\Sigma}$

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

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

Using the block structure, the difference can be efficiently computed as

$$\left|\left|X_{k+1}^{(\ell)}\boldsymbol{\tau}_{k+1}(X_{k+1}^{(r)})^\top - X_k^{(\ell)}\boldsymbol{\tau}_k(X_k^{(r)})^\top\right|\right|_F = ||\boldsymbol{\tau}_{k+1}||_F + ||\boldsymbol{\tau}_k||_F - 2\mathrm{tr}\Big(\boldsymbol{\tau}_{k+1}((X_{k+1}^r)^\top X_{k+1}^{(\ell)})\boldsymbol{\tau}_k((X_k^r)^\top X_k^{(\ell)})\Big)$$

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 `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 mat-form preconditioned CG [Kressner et al. 2011; Simoncini et al. 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 – [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

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

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

Numerical example I – 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 `tol` $= 5 \cdot 10^{-6}$. Truncation tolerance `tolrank`$= 10^{-12}$. $\mathcal{P}_1$: one-term precond, $\mathcal{P}_2$: two-term precond, expensive.

Numerical example II – [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,

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

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 II – 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 `tol` $= 10^{-6}$. Truncation tolerance `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

Davide Palitta, M. Iannacito, and Valeria Simoncini
A subspace-conjugate gradient method for linear matrix equations,
pp. 1-25, Jan 2025. ArXiv 2501.02938