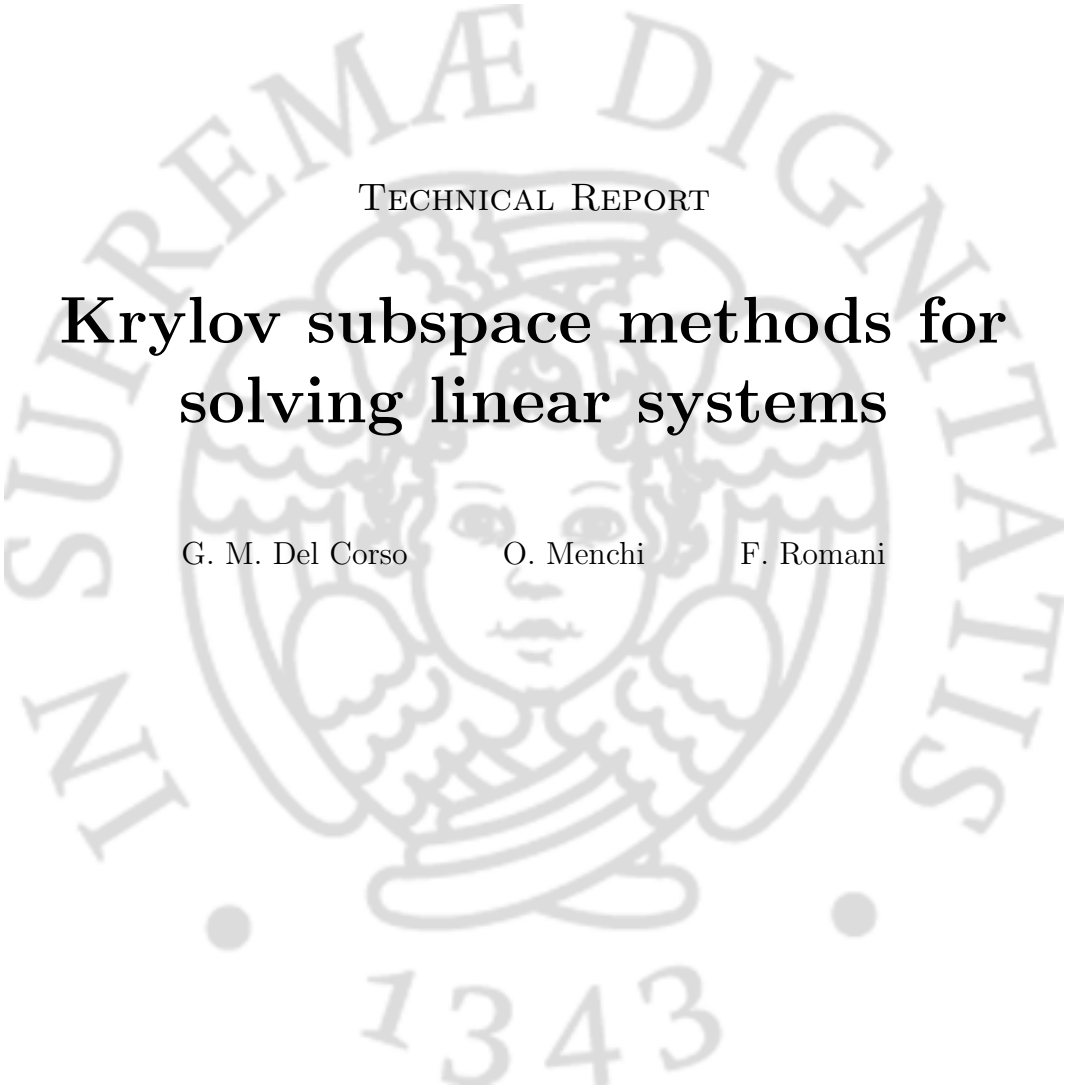
Università di Pisa
Dipartimento di Informatica



Technical Report

# Krylov subspace methods for solving linear systems

G. M. Del Corso O. Menchi F. Romani



Address: Largo B. Pontecorvo 3, 56127 Pisa, Italy. Tel: +39 050 2212700 Fax: +39 050 2212726

# Krylov subspace methods for solving linear systems

G. M. Del Corso O. Menchi F. Romani

## 1 Introduction

With respect to the influence on the development and practice of science and engineering in the 20th century, Krylov subspace methods are considered as one of the most important classes of numerical methods [9].

Given a nonsingular matrix $A \in \mathbf{R}^{N\times N}$ and a vector $\boldsymbol{b} \in \mathbf{R}^N$, consider the system

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

and denote by $\boldsymbol{x}^*$ the solution. When $N$ is large and $A$ does not enjoy particular structure properties, iterative methods are required to solve (1). Most iterative methods start from an initial guess $\boldsymbol{x}_0$ and compute a sequence $\boldsymbol{x}_n$ which approximates $\boldsymbol{x}^*$ moving in an affine subspace $\boldsymbol{x}_0 + \mathcal{K} \subset \mathbf{R}^N$.

To identify suitable subspaces $\mathcal{K}$, standard approaches can be followed. Denoting by $\boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n$ the residual vector at the $n$th iteration step, we can impose orthogonality conditions on $\boldsymbol{r}_n$ or minimize some norm of $\boldsymbol{r}_n$. In any case, we are interested in procedures that allow the construction of $\mathcal{K}$ exploiting only simple operations such as matrix-by-vector products. Using this kind of operations we can construct Krylov subspaces, an ideal setting where to develop iterative methods. Actually, the iterative methods that are today applied for solving large-scale linear systems are mostly Krylov subspace solvers. Classical iterative methods that do not belong to this class, like the successive overrelaxation (SOR) method, are no longer competitive.

The idea of Krylov subspaces iteration was established around the early 1950 by C. Lanczos and W. Arnoldi. Lanczos method, originally applied to the computation of eigenvalues, was based on two mutually orthogonal sequences of vectors and a three-term recurrence. In 1952 M. Hestenes and E. Stiefel gave their classical description of the conjugate gradient method (CG), presented as a direct method, rather than an iterative method. The method was related to the Lanczos method, but reduced the two mutually orthogonal sequences to just one and the three-term recurrence to a couple of two-term recurrences. At first the CG did not receive much attention because of its intrinsic instability, but in 1971 J. Reid pointed out its effectiveness as an iterative method for symmetric positive definite systems. Since then, a considerable part of the research in numerical linear algebra has been devoted to generalizations of CG to nonsymmetric or indefinite systems.

The assumption we have made on the nonsingularity of $A$ greatly simplifies the problem since if $A$ is singular, Krylov subspaces methods can fail. Even if a solution $\boldsymbol{x}^*$ of (1) exists, it may not belong to a Krylov subspace.

## 2 Krylov subspaces

Alexei Krylov was a Russian mathematician who published a paper on the subject in 1931 [20]. The basis for its subspaces can be found in the Cayley-Hamilton theorem, which says that the inverse of a matrix $A$ is expressed in terms of a linear combination of powers of $A$. The Krylov subspace methods share the feature that the matrix $A$ needs only be known as an operator (for example through a subroutine) which gives the matrix-by-vector product $A\boldsymbol{v}$ for any $N$-vector $\boldsymbol{v}$.

Given a vector $\boldsymbol{v} \in \mathbf{R}^N$ and an integer $n \leq N$, a *Krylov subspace* is

$$\mathcal{K}_n = \mathcal{K}_n\big(A, \boldsymbol{v}\big) = \mathrm{span}\big(\boldsymbol{v}, A\,\boldsymbol{v}, A^2\,\boldsymbol{v}, \ldots, A^{n-1}\,\boldsymbol{v}\big),$$

i.e. $\mathcal{K}_n$ is the subspace of all the vectors $\boldsymbol{z}$ of $\mathbf{R}^N$ which can be written in the form

$$\boldsymbol{z} = \pi(A)\,\boldsymbol{v}, \quad \text{with} \quad \pi \in \mathcal{P}_{n-1},$$

where $\mathcal{P}_j$ is the set of all the polynomials of degree $\leq j$.

Clearly, $\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \mathcal{K}_3 \ldots$, and the dimension increases at most by one in each step. It is evident that the dimension cannot exceed $N$, but it can be much smaller. In fact, it is bounded by the *degree* of $\boldsymbol{v}$ with respect to $A$, i. e. the minimal degree $\nu$ of the polynomial $\pi$ such that $\pi(A)\boldsymbol{v} = \boldsymbol{0}$. $\mathcal{K}_\nu$ is invariant and cannot be further enlarged, hence

$$\dim \mathcal{K}_n\big(A, \boldsymbol{v}\big) = \min(n, \nu).$$

The main question is: why a Krylov subspace is a suitable space where to look for an approximate solution of system (1)? The answer is that the solution of a linear system has a natural representation as a member of a Krylov subspace, and if the dimension of this space is small, the solution can be found exactly (or well approximated) in a few iterations.

It is not easy to give a formal definition of Krylov space solvers that covers all relevant cases [16]: a (standard) Krylov subspace solver is an iterative method which starts from an initial approximation $\boldsymbol{x}_0$ and the corresponding residual $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ and generates iterates $\boldsymbol{x}_n$ such that

$$\boldsymbol{x}_n - \boldsymbol{x}_0 = \pi_{n-1}(A)\,\boldsymbol{r}_0, \quad \text{i.e.} \quad \boldsymbol{x}_n \in \boldsymbol{x}_0 + \mathcal{K}_n(A, \boldsymbol{r}_0), \tag{2}$$

with $\pi_{n-1} \in \mathcal{P}_{n-1}$, for all, or at least most $n$, until it possibly finds the exact solution of the system. For some $n$, $\boldsymbol{x}_n$ may not exist or $\pi_{n-1}$ may have lower degree. The residuals $\boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n$ of a Krylov space solver satisfy

$$\boldsymbol{r}_n - \boldsymbol{r}_0 = \xi_n(A)\,\boldsymbol{r}_0 \in A\mathcal{K}_n(A, \boldsymbol{r}_0) \in \mathcal{K}_{n+1}(A, \boldsymbol{r}_0),$$

where $\xi_n \in \mathcal{P}_n$ is related to polynomial $\pi_{n-1}$ by

$$\xi_n(z) = 1 - z\,\pi_{n-1}(z), \quad \text{with} \quad \xi_n(0) = 1.$$

The vague expression "for all, or at least most $n$" used in the definition, is needed because in some widely used Krylov space solvers (e.g. BiCG ) there may exist exceptional situations, where for some $n$ the iterate $\boldsymbol{x}_n$ and the residual $\boldsymbol{r}_n$ are not defined. There are also nonstandard Krylov subspace methods where the space for $\boldsymbol{x}_n - \boldsymbol{x}_0$ is still a Krylov space but one that differs from $\mathcal{K}_n(A, \boldsymbol{r}_0)$.

# 3 The symmetric positive definite (SPD) case

Hence the idea behind Krylov subspace solvers is to generate sequences of approximate solutions $\boldsymbol{x}_n$ in a Krylov subspace converging to $\boldsymbol{x}^*$. Here, convergence may also mean that after $n$ steps $\boldsymbol{x}_n = \boldsymbol{x}^*$ and the process stops (finite termination). This is in particular true (in exact arithmetic) if the method ensures that the corresponding residuals are linearly independent.

All this holds in theory, but a first difficulty arises in practice: the vectors $A^j\,\boldsymbol{v}$, $j = 1, 2, \ldots$, usually become almost linear dependent in a few steps, hence methods relying on Krylov subspaces must involve some orthogonalization scheme to construct bases for the space. One such method, frequently applied, is the Arnoldi orthogonalization algorithm, which implements a Gram-Schmidt technique. This algorithm is suitable for general matrices, but here we describe it in the Lanczos version for symmetric matrices.

### 3.1 The Lanczos algorithm

Starting from a vector $\boldsymbol{v}$ such that $\|\boldsymbol{v}\|_2 = 1$, the algorithm constructs an orthonormal basis for $\mathcal{K}_n\big(A, \boldsymbol{v}\big)$ with $n \leq N$.

---

```
Lanczos algorithm
```
$\boldsymbol{w}_0 = \boldsymbol{0}$
$\boldsymbol{w}_1 = \boldsymbol{v}$
$\delta_1 = 0$
`for` $j = 1, \ldots, n$
$\boldsymbol{h} = A\,\boldsymbol{w}_j - \delta_j\,\boldsymbol{w}_{j-1}$
$\gamma_j = \boldsymbol{h}^T \boldsymbol{w}_j$
$\boldsymbol{k} = \boldsymbol{h} - \gamma_j\,\boldsymbol{w}_j$
$\delta_{j+1} = \|\boldsymbol{k}\|_2$
$\boldsymbol{w}_{j+1} = \boldsymbol{k}/\delta_{j+1}$
`end`

---

In floating point arithmetic Lanczos algorithm can be unstable because cancellation errors may occur. In this case the orthogonality is lost. Stabilizing techniques like restarting, are suggested or more stable algorithms, for example Householder algorithm, can be taken into consideration.

The stopping control $\delta_{j+1} \neq 0$ must be added. If the condition $\delta_{j+1} = 0$ occurs for $j < n$, it means that $\boldsymbol{v}$ has degree $j$, hence $\mathcal{K}_j$ is invariant (this occurrence is called *lucky breakdown*). Otherwise in exact arithmetic an orthonormal basis $W_n$ of $\mathcal{K}_n$ is obtained, with

$$W_n = \big[\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\big] \in \mathbf{R}^{N\times n}, \quad \text{(hence } W_n^T W_n = I\text{)}.$$

Basically, Lanczos algorithm implements the three term recurrent relation

$$\delta_{j+1}\boldsymbol{w}_{j+1} = A\,\boldsymbol{w}_j - \gamma_j \boldsymbol{w}_j - \delta_j \boldsymbol{w}_{j-1},$$

which can be written in the form

$$A\,W_n = W_n\,T_n + \delta_{n+1}\,\boldsymbol{w}_{n+1}\boldsymbol{e}_n^T, \tag{3}$$

where $T_n$ is the symmetric tridiagonal matrix

$$T_n = \begin{bmatrix} \gamma_1 & \delta_2 & & \\ \delta_2 & \gamma_2 & \ddots & \\ & \ddots & \ddots & \delta_n \\ & & \delta_n & \gamma_n \end{bmatrix}. \tag{4}$$

Because of the orthogonality of the vectors $\boldsymbol{w}_j$ we have

$$W_n^T\,A\,W_n = T_n. \tag{5}$$

The eigenvalues $\theta_1^{(n)}, \ldots, \theta_n^{(n)}$ of $T_n$ (called *Ritz values* of $A$) play an important role in the study of the convergence of Krylov subspace methods. Using (5) it can be shown that when $n$ increases the eigenvalues $\theta_j^{(n)}$ approximate eigenvalues of $A$, starting from the largest ones, and the corresponding eigenvectors $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n$ of $T_n$ approximate eigenvectors $W_n\boldsymbol{t}_1, \ldots, W_n\boldsymbol{t}_n$ of $A$.

The characteristic polynomial of $T_n$

$$\pi_n(\lambda) = \det(\lambda I_n - T_n) = \prod_{j=1}^{n}(\lambda - \theta_j^{(n)}) \tag{6}$$

can be computed recursively using the following three term recursion

$$\pi_j(\lambda) = (\lambda - \gamma_j)\pi_{j-1}(\lambda) - \delta_j^2 \pi_{j-2}(\lambda), \quad \pi_0(\lambda) = 1, \quad \pi_1(\lambda) = \lambda - \gamma_1. \tag{7}$$

The following property holds: if $\delta_j \neq 0$ for $j \leq n+1$, all the nonzero vectors of $\mathcal{K}_{n+1}(A, \boldsymbol{v})$ which are orthogonal to $\mathcal{K}_n(A, \boldsymbol{v})$ can be written as $\alpha \pi_n(A)\boldsymbol{v}$ for some $\alpha \neq 0$.

## 3.2 Projections

Denote by $< \cdot, \cdot >_A$ the A-inner product in $\mathbf{R}^N$ and by $\| \cdot \|_A$ the corresponding induced A-norm. Let $\mathcal{K}_n = \mathcal{K}_n(A, \boldsymbol{v})$ be the Krylov subspace generated by a given vector $\boldsymbol{v} \in \mathbf{R}^N$ and denote by $W_n$ an orthonormal basis of $\mathcal{K}_n$. The matrix

$$P_A = W_n (W_n^T A W_n)^{-1} W_n^T A$$

is the *A-orthogonal projector* onto $\mathcal{K}_n$ (in fact, $P_A^2 = P_A$). For any $\boldsymbol{y} \in \mathbf{R}^N$, the vector $P_A \boldsymbol{y}$ is the point of $\mathcal{K}_n$ which realizes the minimum A-distance from $\boldsymbol{y}$, i.e.

$$\|\boldsymbol{y} - P_A \boldsymbol{y}\|_A = \min_{\boldsymbol{z} \in \mathcal{K}_n} \|\boldsymbol{y} - \boldsymbol{z}\|_A.$$

Let $\boldsymbol{y} = \boldsymbol{x}^* - \boldsymbol{x}_0$, then the vector

$$P_A \boldsymbol{y} = W_n (W_n^T A W_n)^{-1} W_n^T A (\boldsymbol{x}^* - \boldsymbol{x}_0) = W_n (W_n^T A W_n)^{-1} W_n^T \boldsymbol{r}_0 \tag{8}$$

minimizes the problem

$$\min_{\boldsymbol{x} \in \boldsymbol{x}_0 + \mathcal{K}_n} \phi(\boldsymbol{x}), \quad \text{where} \quad \phi(\boldsymbol{x}) = \tfrac{1}{2} \|\boldsymbol{x}^* - \boldsymbol{x}\|_A^2. \tag{9}$$

This minimizer is assumed as the $n$th iterate of an orthogonal projection method, by setting $\boldsymbol{x}_n - \boldsymbol{x}_0 = P_A\big(\boldsymbol{x}^* - \boldsymbol{x}_0\big)$. Using (8) we express $\boldsymbol{x}_n - \boldsymbol{x}_0$ in terms of the basis $W_n$ as

$$\boldsymbol{x}_n - \boldsymbol{x}_0 = W_n \boldsymbol{c}_n, \quad \text{where} \quad \boldsymbol{c}_n = (W_n^T A W_n)^{-1} W_n^T \boldsymbol{r}_0. \tag{10}$$

We consider now the special Krylov subspace $\mathcal{K}_n = \mathcal{K}_n(A, \boldsymbol{r}_0)$, whose orthonormal basis $W_n$ can be constructed by applying Lanczos algorithm with $\boldsymbol{v} = \boldsymbol{r}_0 / \|\boldsymbol{r}_0\|_2$. The orthogonal projection method whose $n$th iterate $\boldsymbol{x}_n$ minimizes $\phi(\boldsymbol{x})$ onto the affine space $\boldsymbol{x}_0 + \mathcal{K}_n\big(A, \boldsymbol{r}_0\big)$, i.e.

$$\|\boldsymbol{x}^* - \boldsymbol{x}_n\|_A = \min_{\boldsymbol{x} \in \mathcal{K}_n} \|\boldsymbol{x}^* - \boldsymbol{x}_0 - \boldsymbol{x}\|_A,$$

is our Krylov subspace method [27]. Denoting by $\boldsymbol{\epsilon}_n = \boldsymbol{x}^* - \boldsymbol{x}_n$ the $n$th error, we have

$$\begin{aligned} \|\boldsymbol{\epsilon}_n\|_A &= \min_{\pi \ \in \mathcal{P}_{n-1}} \|\boldsymbol{\epsilon}_0 - \pi(A)\, \boldsymbol{r}_0\|_A = \min_{\pi \ \in \mathcal{P}_{n-1}} \|\boldsymbol{\epsilon}_0 - A\, \pi(A)\, \boldsymbol{\epsilon}_0\|_A \\ &= \min_{\xi \ \in \mathcal{P}_n,\ \xi(0) = 1} \|\xi(A)\, \boldsymbol{\epsilon}_0\|_A. \end{aligned} \tag{11}$$

The matrix $P_A$ does not have to be formed explicitly, since it is available as a by-product of the algorithm.

Comparing (10) and (5) we get

$$\boldsymbol{x}_n - \boldsymbol{x}_0 = \boldsymbol{c}_n, \quad \text{where} \quad \boldsymbol{c}_n = T_n^{-1} W_n^T \boldsymbol{r}_0. \tag{12}$$

The columns of $W_n$ are orthonormal, then

$$W_n^T \boldsymbol{r}_0 = W_n^T \|\boldsymbol{r}_0\|_2 \boldsymbol{v} = \|\boldsymbol{r}_0\|_2\, \boldsymbol{e}_1,$$

and

$$W_n T_n \boldsymbol{c}_n = W_n W_n^T \boldsymbol{r}_0 = \|\boldsymbol{r}_0\|_2\, W_n\, \boldsymbol{e}_1 = \|\boldsymbol{r}_0\|_2\, \boldsymbol{v} = \boldsymbol{r}_0. \tag{13}$$

From (10) and (3) we get

$$\boldsymbol{r}_n - \boldsymbol{r}_0 = A\,(\boldsymbol{x}_0 - \boldsymbol{x}_n) = -AW_n\boldsymbol{c}_n = -W_n\,T_n\boldsymbol{c}_n - \delta_{n+1}\boldsymbol{e}_n^T\boldsymbol{c}_n\,\boldsymbol{w}_{n+1}.$$

Using (13) we get

$$\boldsymbol{r}_n = -\delta_{n+1}\,\boldsymbol{e}_n^T\,\boldsymbol{c}_n\,\boldsymbol{w}_{n+1}, \tag{14}$$

showing that the residuals $\boldsymbol{r}_n$ are multiple of the $\boldsymbol{w}_{n+1}$, hence they are orthogonal.

The matrix $T_n$ in (4) is SPD. By the Choleski factorization we can write $T_n = L_nL_n^T$, where $L_n$ is a lower bidiagonal matrix. Denote by $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ the columns of the matrix $S_n = W_nL_n^{-T}$. Since

$$S_n^TAS_n = L_n^{-1}W_n^TAW_nL_n^{-T} = L_n^{-1}T_nL_n^{-T} = I,$$

the vectors $\boldsymbol{s}_j$ are $A$-conjugate and $S_n$ results an $A$-conjugate basis for $\mathcal{K}_n$.

Due to the structure of $L_n^{-T}$, the vector $\boldsymbol{s}_n$ is given by a linear combination of $\boldsymbol{w}_n$ and $\boldsymbol{s}_{n-1}$. From (12) we get

$$\boldsymbol{x}_n - \boldsymbol{x}_0 = S_n\,\boldsymbol{z}_n, \quad \text{where} \quad \boldsymbol{z}_n = L_n^{-1}\,\|\boldsymbol{r}_0\|_2\,\boldsymbol{e}_1,$$

and analogously

$$\boldsymbol{x}_{n-1} - \boldsymbol{x}_0 = S_{n-1}\,\boldsymbol{z}_{n-1}.$$

It is easy to verify that

$$\boldsymbol{z}_n = \begin{bmatrix} \boldsymbol{z}_{n-1} \\ \zeta_n \end{bmatrix}$$

for a suitable $\zeta_n$. Hence the following recursion holds

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \zeta_n\,\boldsymbol{s}_n. \tag{15}$$

### 3.3 Conjugate gradient algorithm (CG)

The conjugate gradient (CG) is due to Hestenes and Stiefel [19]. It is applied to symmetric positive definite (SPD) matrices, and is still the method of choice for this case.

Relation (15), appropriately rewritten, gives the basis for CG. For $n \geq 1$ let $\boldsymbol{p}_n$ be a multiple of $\boldsymbol{s}_{n+1}$. Then $\boldsymbol{p}_{n+1}$ is given by a combination of $\boldsymbol{p}_n$ and $\boldsymbol{w}_{n+1}$, which by (14) is a multiple of $\boldsymbol{r}_n$. Then we set

$$\boldsymbol{p}_{n+1} = \boldsymbol{r}_{n+1} + \beta_n\boldsymbol{p}_n, \tag{16}$$

for suitable scalars $\beta_n$. The vector $\boldsymbol{p}_n$ is called the *search direction*. From (15) we can express

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n\,\boldsymbol{p}_n, \tag{17}$$

for suitable scalars $\alpha_n$. The corresponding residuals must satisfy

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n\,A\,\boldsymbol{p}_n. \tag{18}$$

Requiring that $\boldsymbol{r}_{n+1}$ be orthogonal to $\boldsymbol{r}_n$ and $\boldsymbol{p}_{n+1}$ be A-conjugate to $\boldsymbol{p}_n$ we get

$$\alpha_n = \frac{< \boldsymbol{r}_n, \boldsymbol{r}_n >}{< A\boldsymbol{p}_n, \boldsymbol{r}_n >}, \tag{19}$$

and

$$\beta_n = -\frac{< \boldsymbol{r}_{n+1}, A\boldsymbol{p}_n >}{< \boldsymbol{p}_n, A\boldsymbol{p}_n >}. \tag{20}$$

It is easy to prove by induction that

$$< \boldsymbol{r}_i, \boldsymbol{r}_n >= 0, \quad < \boldsymbol{p}_i, \boldsymbol{p}_n >_A = 0, \quad \text{for} \quad i \neq n,$$

(i.e. all the residuals are orthogonal and all the vectors $\boldsymbol{p}_n$ are A-conjugate) and

$$< \boldsymbol{p}_i, \boldsymbol{r}_n >= \begin{cases} 0 & \text{for } i \leq n-1, \\ \|\boldsymbol{r}_i\|_2^2 & \text{for } i \geq n. \end{cases} \tag{21}$$

Moreover

$$< A\boldsymbol{p}_n, \boldsymbol{r}_n >=< A\boldsymbol{p}_n, \boldsymbol{p}_n - \beta_{n-1}\boldsymbol{p}_{n-1} >=< A\boldsymbol{p}_n, \boldsymbol{p}_n >,$$

giving the following alternative forms for $\alpha_n$

$$\alpha_n = \frac{< \boldsymbol{p}_n, \boldsymbol{r}_n >}{< A\boldsymbol{p}_n, \boldsymbol{p}_n >}.$$

Hence $\alpha_n$ satisfies the minimum problem

$$\phi(\boldsymbol{x}_{n+1}) = \phi(\boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n) = \min_{\alpha} \ \phi(\boldsymbol{x}_n + \alpha \boldsymbol{p}_n)$$

on the space $\boldsymbol{x}_0 + \mathcal{K}_n(A, \boldsymbol{r}_0)$, where $\phi$ is the function defined in (9). It follows that the sequence

$$\phi(\boldsymbol{x}_n) = \tfrac{1}{2} \|\boldsymbol{x}^* - \boldsymbol{x}_n\|_A^2 = \tfrac{1}{2} \left(\boldsymbol{x}^* - \boldsymbol{x}_n\right)^T \boldsymbol{r}_n, \quad n = 0, 1, \ldots$$

is nonincreasing. An alternative form for $\beta_n$ can be obtained by noticing that

$$\begin{aligned} < \boldsymbol{r}_{n+1}, \boldsymbol{r}_{n+1} >=< \boldsymbol{r}_{n+1}, \boldsymbol{r}_n - \alpha_n A\, \boldsymbol{p}_n >= -\alpha_n \ < \boldsymbol{r}_{n+1}, A\, \boldsymbol{p}_n > \\ = -\frac{< \boldsymbol{r}_n, \boldsymbol{r}_n >}{< A\boldsymbol{p}_n, \boldsymbol{p}_n >} \ < \boldsymbol{r}_{n+1}, A\, \boldsymbol{p}_n >= \beta_n \ < \boldsymbol{r}_n, \boldsymbol{r}_n >, \end{aligned}$$

hence

$$\beta_n = \frac{< \boldsymbol{r}_{n+1}, \boldsymbol{r}_{n+1} >}{< \boldsymbol{r}_n, \boldsymbol{r}_n >}.$$

---

`CG algorithm`
$\boldsymbol{x}_0$ a starting vector. Often $\boldsymbol{x}_0 = \boldsymbol{0}$
$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
$\boldsymbol{p}_0 = \boldsymbol{r}_0$
`for` $n = 0, 1, \ldots$ until convergence
$\alpha_n = \|\boldsymbol{r}_n\|_2^2 / \left(\boldsymbol{r}_n^T A \boldsymbol{p}_n\right)$
$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n$
$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n A\, \boldsymbol{p}_n$
$\beta_n = \|\boldsymbol{r}_{n+1}\|_2^2 / \|\boldsymbol{r}_n\|_2^2$
$\boldsymbol{p}_{n+1} = \boldsymbol{r}_{n+1} + \beta_n \boldsymbol{p}_n$
`end` (a control on $\boldsymbol{r}_n^T A \boldsymbol{p}_n \neq 0$ must be provided).

---

In this code, the residual $\boldsymbol{r}_n$ is computed according to recursion (18), but this formula is more prone to instability than the definition $\boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n$. If (18) is used to keep low the computational cost per iteration, a control on the orthogonality of the residuals should be performed now and then.

A three term recurrence for $\boldsymbol{r}_n$ is easily obtained:

$$\begin{aligned} \boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n A\, \boldsymbol{p}_n = (I - \alpha_n A)\boldsymbol{r}_n - \alpha_n \beta_{n-1} A\, \boldsymbol{p}_{n-1} \\ = \Big(\tau_n I - \alpha_n A\Big)\boldsymbol{r}_n + (1 - \tau_n)\boldsymbol{r}_{n-1}, \end{aligned} \tag{22}$$

where

$$\tau_0 = 1, \quad \tau_n = 1 + \frac{\alpha_n \beta_{n-1}}{\alpha_{n-1}} \quad \text{for} \quad n \geq 1.$$

An analogous recurrence holds for $\boldsymbol{x}_n$.

The coefficients $\gamma_j$ and $\delta_j$ of the Lanczos algorithm can be easily derived from the coefficients $\alpha_j$ and $\beta_j$ of CG. In fact, exploiting the facts that the columns $\boldsymbol{w}_j$ of $W_n$ are normalized and the vectors $\boldsymbol{p}_j$ are A-conjugate, it is possible to show that

$$\gamma_1 = \frac{1}{\alpha_0}, \quad \gamma_{j+1} = \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}}, \quad \delta_{j+1} = \frac{\sqrt{\beta_{j-1}}}{\alpha_{j-1}}, \quad \text{for} \quad j \geq 1.$$

Using formulas (16) – (21) we get

$$\boldsymbol{x}_j - \boldsymbol{x}_0 = \sum_{i=0}^{j-1} \alpha_i \boldsymbol{p}_i = 2 \sum_{k=0}^{j-1} \frac{\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}_j)}{\|\boldsymbol{r}_k\|_2^2} \, \boldsymbol{r}_k.$$

In particular, if $\boldsymbol{x}_0 = \boldsymbol{0}$

$$\|\boldsymbol{x}_j\|_2^2 = 2 \sum_{k=0}^{j-1} \frac{\big(\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}_j)\big)^2}{\|\boldsymbol{r}_k\|_2^2}.$$

Since $\phi(\boldsymbol{x}_k) \geq \phi(\boldsymbol{x}_j)$ for $k < j$ and $\phi(\boldsymbol{x}_j)$ decreases with $j$, the terms in the sum increase with $j$. It follows that $\|\boldsymbol{x}_j\|_2$ increases monotonically with $j$.

In exact arithmetic, from the orthogonality of the residuals it follows that $\boldsymbol{r}_n = \boldsymbol{0}$ for an index $n \leq N$, i.e. CG is finite. In floating point arithmetic this is hardly true, so CG is applied as an iterative method, whose convergence is worth studying.

Since CG is an orthogonal projection method on the Krylov subspace $\mathcal{K}_n(A, \boldsymbol{r}_0)$, by (11) the $n$th iterate verifies

$$\|\boldsymbol{\epsilon}_n\|_A^2 = \|\boldsymbol{x}^* - \boldsymbol{x}_n\|_A^2 = \min \|\xi(A)\boldsymbol{\epsilon}_0\|_A^2, \tag{23}$$

on the polynomials $\xi$ of degree $\leq n$ such that $\xi(0) = 1$. Let $\lambda_i$, $i = 1, \ldots, N$, be the eigenvalues of $A$ and $\boldsymbol{u}_i$ the corresponding normalized eigenvectors. We can express $\boldsymbol{\epsilon}_0$ in terms of this basis in the form

$$\boldsymbol{\epsilon}_0 = \sum_{i \in \mathcal{I}} \tau_i \, \boldsymbol{u}_i, \tag{24}$$

where $\mathcal{I}$ is a suitable subset of $\{1, \ldots, N\}$, then

$$\|\boldsymbol{\epsilon}_0\|_A^2 = \sum_{i \in \mathcal{I}} \lambda_i \tau_i^2$$

and

$$\|\xi(A)\boldsymbol{\epsilon}_0\|_A^2 = \|\sum_{i \in \mathcal{I}} \xi(\lambda_i)\tau_i \, \boldsymbol{u}_i\|_A^2 = \sum_{i \in \mathcal{I}} \lambda_i \xi^2(\lambda_i) \, \tau_i^2 \leq \max_{i \in \mathcal{I}} \xi^2(\lambda_i) \, \|\boldsymbol{\epsilon}_0\|_A^2.$$

This bound involves only the eigenvectors that effectively appear in expression (24) of $\boldsymbol{\epsilon}_0$. Actually, in floating point arithmetic the eigenvalues which do not appear in (24) may be reintroduced by the accumulation of the round-off errors. Hence we will consider the set $\Lambda$ of the eigenvalues of the whole matrix $A$, hence

$$\max_{i \in \mathcal{I}} \xi^2(\lambda_i) \leq \max_{\lambda \in \Lambda} \xi^2(\lambda). \tag{25}$$

From (25) it follows that

$$\|\boldsymbol{\epsilon}_n\|_A = \|\boldsymbol{x}^* - \boldsymbol{x}_n\|_A = \min_{\xi \in \mathcal{P}_n} \max_{\lambda \in \Lambda} |\xi(\lambda)| \, \|\boldsymbol{\epsilon}_0\|_A. \tag{26}$$

Using the fact that the minimax polynomial of degree $n$ which solves (26) is a shifted and scaled Chebyshev polynomial of the first kind, we obtain the bound

$$\|\boldsymbol{\epsilon}_n\|_A \leq 2 \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^n \|\boldsymbol{\epsilon}_0\|_A, \tag{27}$$

where $\kappa = \lambda_{\max}/\lambda_{\min}$ is the condition number of $A$.

In some situations it has been observed that during the iteration the rate of convergence tends to increase, giving an almost superlinear convergence. This property follows from the convergence of the Ritz values of $A$.

If $N$ is large and $A$ is ill-conditioned, the number of iterations necessary to achieve an acceptable approximation may be very large. If this is the case, one has to resort to preconditioning, that is to apply some modification to the original system to get an easier problem. Here "easier" means that the looked for solution can be approximated with a lower computational effort. In this paper we will not insist on this issue.

# 4 The nonsymmetric case

Thanks to the success of CG in the SPD case, much effort has been devoted to its generalizations to the nonSPD case. A straightforward extension is to apply CG to one of the SPD systems

$$A^T A\boldsymbol{x} = A^T\boldsymbol{b}. \tag{28}$$

and

$$AA^T\boldsymbol{y} = \boldsymbol{b}, \quad \boldsymbol{x} = A^T\boldsymbol{y}. \tag{29}$$

The system in (28) is referred as normal equations and the corresponding method is called CGNR. The second approach leads to a method called CGNE. The convergence rate of both algorithms is generally unsatisfactory when $A$ is ill-conditioned.

## 4.1 The CGNR algorithm

When applying CG to the normal equations (28) it is possible to avoid the explicit construction of the matrix $A^TA$ using auxiliary vectors $\boldsymbol{z}_j = A\boldsymbol{p}_j$ and $\boldsymbol{q}_j = A^T\boldsymbol{r}_j$ and noticing that

$$< A^TA\boldsymbol{p}_j, \boldsymbol{p}_j > = < A\boldsymbol{p}_j, A\boldsymbol{p}_j > = < \boldsymbol{z}_j, \boldsymbol{z}_j > .$$

The following algorithm computes the residual $\boldsymbol{q}_j = A^T(\boldsymbol{b} - A\boldsymbol{x}_j)$ relative to system (28) instead of the residual $\boldsymbol{r}_j = \boldsymbol{b} - A\boldsymbol{x}_j$ relative to system (1). It does not compute explicitly $\boldsymbol{r}_j$, which if required, must be computed directly.

The projection is performed onto the Krylov subspace $\mathcal{K}_n = \mathcal{K}_n(A^TA, \boldsymbol{q}_0) = \mathcal{K}_n(A^TA, A^T\boldsymbol{r}_0)$.

---

`CGNR algorithm`

$\boldsymbol{x}_0$ a starting vector
$\boldsymbol{q}_0 = A^T\big(\boldsymbol{b} - A\boldsymbol{x}_0\big)$
$\boldsymbol{p}_0 = \boldsymbol{q}_0$
`for` $j = 0, 1, \ldots$ until convergence
  $\boldsymbol{z}_j = A\boldsymbol{p}_j$
  $\alpha_j = \|\boldsymbol{q}_j\|_2^2 / \|\boldsymbol{z}_j\|_2^2$
  $\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j\,\boldsymbol{p}_j$
  $\boldsymbol{q}_{j+1} = \boldsymbol{q}_j - \alpha_j\,A^T\,\boldsymbol{z}_j$
  $\beta_j = \|\boldsymbol{q}_{j+1}\|_2^2 / \|\boldsymbol{q}_j\|_2^2$
  $\boldsymbol{p}_{j+1} = \boldsymbol{q}_{j+1} + \beta_j\boldsymbol{p}_j$
`end`

---

Rewriting the properties of CG seen in Section 3.3 with the matrix $A$ replaced by $A^TA$, we see that the vectors $\boldsymbol{q}_j$ are orthogonal, the vectors $\boldsymbol{p}_j$ are $A^TA$-conjugate, hence also the vectors $\boldsymbol{z}_j$ are orthogonal. At each step the difference $\boldsymbol{x}_j - \boldsymbol{x}_0$ minimizes the function

$$\phi(\boldsymbol{x}) = \tfrac{1}{2}|\boldsymbol{x}^* - \boldsymbol{x}\|^2_{A^TA} = \tfrac{1}{2}\,\|A(\boldsymbol{x}^* - \boldsymbol{x})\|_2^2 = \tfrac{1}{2}\,\|\boldsymbol{b} - A\boldsymbol{x}\|_2^2.$$

Hence $\boldsymbol{x}_j - \boldsymbol{x}_0$ is the point which realizes the minimal norm of the residual $\boldsymbol{r}_j$ in $\mathcal{K}_j$, giving a nonincreasing sequence of $\|\boldsymbol{r}_j\|_2$, $j = 0, 1, \ldots$ Moreover, as in the case of an SPD matrix, if $\boldsymbol{x}_0 = \boldsymbol{0}$, the norms $\|\boldsymbol{x}_j\|_2$ increase monotonically with $j$.

For the convergence of CGNR, (27) now becomes

$$\|\boldsymbol{\epsilon}_n\|_{A^TA} = \|\boldsymbol{r}_n\|_2 \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n \|\boldsymbol{r}_0\|_2, \quad \text{where} \quad \kappa = \frac{\sigma_1^2}{\sigma_n^2},$$

the $\sigma_j$ being the singular values of $A$. If $A$ is ill-conditioned, the system of the normal equations is much worse conditioned, since $\kappa$ for $A^TA$ is the square of $\kappa$ for $A$.

A reorganization of CG similar to the one used to get CGNR leads to CGNE. The method works in the affine space $\boldsymbol{x}_0 + A^T\mathcal{K}_n(AA^T, \boldsymbol{r}_0)$, but

$$A^T\mathcal{K}_n(AA^T, \boldsymbol{r}_0) = \mathcal{K}_n(A^TA, A^T\boldsymbol{r}_0),$$

hence the two methods essentially work in the same space. The difference is that CGNR achieves optimality for the residuals, while CGNE achieves optimality for the errors.

## 4.2 Other CG-type algorithms

Due to the unsatisfactory convergence behavior of CGNR (and of CGNE for the same reason) when $A$ is ill-conditioned, other CG-type algorithms have been proposed, with better convergence properties.

An extension of CG to the non SPD case should maintain at least one of its feature, either minimize the error function $\Phi(\boldsymbol{x})$ on the Krylov subspace, or satisfy some orthogonal condition on the residuals. These two features, which in the SPD case are satisfied in the same time, in the general case are not and must be addressed separately [14]. Hence there are two different ways to generalize CG :

- Maintain the orthogonality of the projection by constructing either orthogonal residuals $\boldsymbol{r}_n$ or $A^TA$-orthogonal search directions. Then, the recursions involve all previously constructed residuals or search directions and all previously constructed iterates.
- Maintain short recurrence formulas for residuals, direction vectors and iterates. The resulting methods are at best oblique projection methods. There is no minimality property of error or residuals vectors.

Since Krylov subspace methods for solving nonsymmetric linear systems represent an active field of research, new methods are continuously proposed. We give here a brief description of the most popular ones, taken from [1]. The common denominator of all these methods is to provide acceptable solutions in a number of iterations much less than the order of $A$. Naturally this number may vary much if preconditioning is applied. The different algorithms we consider require different preconditioners to become effective, so we will not go deeper into this subject.

### 4.2.1 Minimal Residual (MINRES) and Symmetric LQ (SYMMLQ)

The MINRES and SYMMLQ methods are variants of the Lanczos method suitable to symmetric indefinite systems. Since they avoid the factorization of the tridiagonal matrix, they do not suffer from breakdowns. MINRES minimizes the residual in the 2-norm. SYMMLQ solves the projected system, but does not minimize anything (it keeps the residual orthogonal to all previous ones).

When $A$ is symmetric but not positive definite, we can still construct an orthogonal basis for the Krylov subspace by a three term recurrence relation of the form

$$A\boldsymbol{r}_j = \boldsymbol{r}_{j+1}t_{j+1,j} + \boldsymbol{r}_j t_{j,j} + \boldsymbol{r}_{j-1}t_{j-1,j},$$

or, in the matrix form, $AR_n = R_{n+1}T_n$, where $T_n$ is an $(n+1) \times n$ tridiagonal matrix. In this case $< \cdot, \cdot >_A$ no longer defines an inner product. However we can still try to minimize the residual in the 2-norm by obtaining

$$\boldsymbol{x}_n \in \mathcal{K}_n(A, \boldsymbol{r}_0), \quad \text{with} \quad \boldsymbol{x}_n = R_n\,\boldsymbol{y},$$

that minimizes

$$\|A\boldsymbol{x}_n - \boldsymbol{b}\|_2 = \|AR_n\,\boldsymbol{y} - \boldsymbol{b}\|_2 = \|R_{n+1}T_n\,\boldsymbol{y} - \boldsymbol{b}\|_2.$$

Now we exploit the fact that if $D_{n+1}$ =diag$(\|\boldsymbol{r}_0\|_2, \ldots, \|\boldsymbol{r}_n\|_2)$, then $R_{n+1}D_{n+1}^{-1}$ is an orthonormal transformation with respect to the current Krylov subspace, and

$$\|A\boldsymbol{x}_n - \boldsymbol{b}\|_2 = \|D_{n+1}T_n\,\boldsymbol{y} - \|\boldsymbol{r}_0\|_2\boldsymbol{e}_1\|_2.$$

This final expression can simply be seen as a minimum norm least squares problem. By applying Givens rotations we obtain an upper bidiagonal system that can simply be solved.

Another possibility is to solve the system $\overline{T}_n\boldsymbol{y} = \|\boldsymbol{r}_0\|_2\boldsymbol{e}_1$, as in the CG method ($\overline{T}_n$ is the upper $n \times n$ part of $T_n$), but here we cannot rely on the existence of a Cholesky decomposition (since $A$ is not SPD). Then we apply an LQ-decomposition. This again leads to simple recurrences and the resulting method is known as SYMMLQ.

### 4.2.2 The Generalized Minimal Residual (GMRES) method

The Generalized Minimal Residual method is an extension of MINRES (which is only applicable to symmetric systems) to unsymmetric systems. Like MINRES, it generates a sequence of orthogonal vectors, but in the absence of symmetry this can no longer be done with short recurrences; instead, all previously computed vectors in the orthogonal sequence have to be retained. For this reason, typically restarted versions of the method are used.

While in CG an orthogonal basis for span$, (\boldsymbol{r}_0, A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, \ldots)$ is formed by the residuals, in GMRES the basis is formed explicitly by applying Gram-Schmidt orthogonalization

$\boldsymbol{w}_n = A\boldsymbol{v}_n$
`for` $j = 1, \ldots, n$
$\boldsymbol{w}_n = \boldsymbol{w}_n - (\boldsymbol{w}_n, \boldsymbol{v}_j)\boldsymbol{v}_j$
`end`
$\boldsymbol{v}_{n+1} = \boldsymbol{w}_n / \|\boldsymbol{w}_n\|_2$

Applied to the Krylov sequence, this orthogonalization is called Arnoldi method. The inner product coefficients $(\boldsymbol{w}_n, \boldsymbol{v}_j)$ and $\|\boldsymbol{w}_n\|_2$ are stored in an upper Hessenberg matrix. The GMRES iterates are constructed as

$$\boldsymbol{x}_n = \boldsymbol{x}_0 + \boldsymbol{y}_1\boldsymbol{v}_1 + \ldots + \boldsymbol{y}_n\boldsymbol{v}_n,$$

where the coefficients have been chosen to minimize the residual norm $\|\boldsymbol{b} - A\boldsymbol{x}_n\|_2$. The GMRES algorithm has the property that this residual norm can be computed without the iterate having been formed. Thus, the expensive action of forming the iterate is not required until the residual norm becomes small enough.

The storage requirements are controlled by restarts. If no restart is performed, GMRES (like any orthogonalizing Krylov subspace method) converges in no more than $N$ steps (assuming exact arithmetic). Of course this is of no practical value when $N$ is large; moreover, the storage and computational requirements in the absence of restarts are prohibitive. Indeed, the crucial element for successful application of GMRES revolves around the decision of when to restart.

Unfortunately, there exist examples for which the method stagnates and convergence takes place only at the $N$th step.

Several suggestions have been made to reduce the increase in memory and computational costs in GMRES. Besides the obvious one of restarting, other approaches include restricting the GMRES search to suitable subspaces of some higher-dimensional Krylov subspace. Methods based on this idea can be viewed as preconditioned GMRES methods. All these approaches have advantages for some problems, but it is not clear a priori which strategy is preferable in any given case.

### 4.2.3 The Biconjugate Gradient (BiCG) method

BiCG goes back to Lanczos [21], but was brought to its current, CG-like form later. CG is not suitable for nonsymmetric systems because the residual vectors cannot be made orthogonal with short recurrences. While GMRES retains orthogonality of the residuals by using long recurrences with a large storage demand, BiCG replaces the orthogonal sequence of residuals by two mutually orthogonal sequences with no minimization.

Together with the orthogonal residuals $\boldsymbol{r}_n \in \mathcal{K}_{n+1}(A, \boldsymbol{r}_0)$ of CG, BiCG constructs a second set of residuals $\widehat{\boldsymbol{r}}_n \in \mathcal{K}_{n+1}(A^T, \widehat{\boldsymbol{r}}_0)$, where the initial residual $\widehat{\boldsymbol{r}}_0$ can be chosen freely. So, BiCG requires two matrix-by-vector products by $A$ and by $A^T$, but there are still short recurrences for $\boldsymbol{x}_n$, $\boldsymbol{r}_n$, and $\widehat{\boldsymbol{r}}_n$. The residuals $\boldsymbol{r}_n$ and $\widehat{\boldsymbol{r}}_n$ form biorthogonal bases for $\mathcal{K}_{n+1}(A, \boldsymbol{r}_0)$ and $\mathcal{K}_{n+1}(A^T, \widehat{\boldsymbol{r}}_0)$, i.e. $\boldsymbol{r}_i^T \widehat{\boldsymbol{r}}_j = 0$ if $i \neq j$. The corresponding directions $\boldsymbol{v}_n$ and $\widehat{\boldsymbol{v}}_n$ form biconjugate bases for $\mathcal{K}_{n+1}(A, \boldsymbol{r}_0)$ and $\mathcal{K}_{n+1}(A^T, \widehat{\boldsymbol{r}}_0)$, i.e. $\boldsymbol{v}_i^T A \widehat{\boldsymbol{v}}_j = 0$ if $i \neq j$.

Few theoretical results are known about the convergence of BiCG. For symmetric positive definite systems the method would give the same results as CG at twice the cost per iteration. For nonsymmetric matrices it has been shown that in the phases of the process where there is a significant reduction of the norm of the residual, BiCG is comparable to GMRES (in terms of numbers of iterations). In practice this is often confirmed. The convergence behavior may be quite irregular, and the method may even break down. Look-ahead strategies can be applied to cope with such situations. Breakdowns can be satisfactorily avoided by restarts. Recent work has focused on avoiding breakdown, on avoiding the use of the transpose and on getting a smoother convergence.

It is difficult to make a fair comparison between GMRES and BiCG. While GMRES minimizes a residual at the cost of increasing work for keeping all the residuals orthogonal and increasing demands for memory space, BiCG does not minimize a residual, but often its accuracy is comparable to GMRES, at the cost of twice the amount of matrix-by-vector products per iteration step. However, the generation of the basis vectors is relatively cheap and the memory requirements are modest. The following variants CGS and Bi-CGSTAB of BiCG have been proposed to increase the effectiveness in certain circumstances.

### 4.2.4 The Conjugate Gradient Squared (CGS) method

CGS, proposed by Sonneveld [28], replaces the multiplication with $A^T$ in BiCG by a second one with $A$. At each iteration step the dimension of the Krylov subspace is increased by two. Convergence is nearly twice as fast as for BiCG, but even more erratic. In circumstances where the computation with $A^T$ is impractical, CGS may be attractive.

### 4.2.5 BiConjugate Gradient Stabilized method (Bi-CGSTAB) method

Bi-CGSTAB was developed by Van der Vorst [29] to avoid the often irregular convergence of CGS, by combining the CGS sequence with a steepest descent update, performing some local optimization and smoothing. Bi-CGSTAB often converges about as fast as CGS, sometimes faster and sometimes not. On the other hand, sometimes the Krylov subspace is not expanded and the method breaks down.

CGS and Bi-CGSTAB are often the most efficient solvers. They have short recurrences, they are typically about twice as fast as BiCG , and they do not require $A^T$. Unlike in GMRES, the memory needed does not increase with the iterations.

#### 4.2.6 The Quasi-Minimal Residual (QMR) method

BiCG often displays rather irregular convergence behavior and breakdown due to the implicit decomposition of the reduced tridiagonal system. To overcome these problems, the Quasi-Minimal Residual method have been proposed by Freund and Nachtigal [13]. The main idea behind this algorithm is to solve the reduced tridiagonal system in a least squares sense, similar to the approach followed by GMRES. Since the constructed basis for the Krylov subspace is bi-orthogonal, rather than orthogonal as in GMRES, the obtained solution is viewed as a quasi-minimal residual solution, which explains the name. Look-ahead techniques to avoid breakdowns in the underlying Lanczos process, which makes QMR more robust than BiCG. The convergence behavior of QMR is much smoother than for BiCG. QMR is expected to converge about as fast as GMRES. When BiCG would make significant progress, QMR arrives at about the same approximation and when BiCG would make no progress at all, QMR still shows slow convergence.

In addition to Bi-CGSTAB, other recent methods have been designed to smooth the convergence of CGS. One idea is to use the quasi-minimal residual (QMR) principle to obtain smoothed iterates from the Krylov subspace generated by other product methods. Such a QMR version of CGS, called TFQMR has been proposed. Numerical experiments show that TFQMR in most cases retains the desirable convergence features of CGS while correcting its erratic behavior. The transpose free nature of TFQMR, its low computational cost and its smooth convergence behavior make it an attractive alternative to CGS.

#### 4.2.7 General remarks

In [1] the storage requirements for the previously described methods are listed, together with their characteristics. In general, the convergence properties of these methods are not well understood. This is particularly true for the transpose-free methods, that have more numerical problems than the corresponding methods which use $A^T$.

Going into details, with GMRES restarts are necessary, resulting in a slower convergence. Then GMRES and related algorithms requires a very effective preconditioner, which in turn might be expensive. On the contrary, the Lanczos-based methods require little work and storage per step, so that the importance of the preconditioners has decreased. But Lanczos scheme in the nonsymmetric case may suffer breakdowns. Hence an algorithm relying on it requires some special look-ahead techniques, to prevent breakdowns.

At the present time we must conclude that there is no clear best overall Krylov subspace method. Each method is a winner in a specific problem class, and the main problem is to identify these classes and to construct new methods for uncovered classes. Among the methods outlined above, there is a class of problems for which a given method is the winner and another one is the loser. Moreover, iterative methods will never reach the robustness of direct methods, nor will they beat direct methods for all the problems.

## 5 Applications: the regularization

When system (1) represents a discrete model describing an underlying continuous phenomenon, the right-hand side $\boldsymbol{b}$ may be contaminated by some kind of noise $\boldsymbol{\eta}$, frequently to due to measurement errors and to the discretization process, i.e.

$$\boldsymbol{b} = \boldsymbol{b}^* + \boldsymbol{\eta}.$$

The vectors $\boldsymbol{b}^*$ and $\boldsymbol{x}^*$ such that $A\boldsymbol{x}^* = \boldsymbol{b}^*$ are considered the exact right-hand side and the exact solution of the system. If the matrix $A$ is ill-conditioned, the solution $\widetilde{\boldsymbol{x}} = A^{-1}\boldsymbol{b}$ may be a

poor approximation of the solution $\boldsymbol{x}^*$, even if the magnitude of $\boldsymbol{\eta}$ is small, and the problem of finding a good approximation of $\boldsymbol{x}^*$ turns out to be a discrete ill-posed problem [18].

To explain of the difficulty of the problem, we express $\widetilde{\boldsymbol{x}}$ through the singular value decomposition (SVD) of $A$. Let $A = U\Sigma V^T$ be the SVD of $A$, where $U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N] \in \mathbf{R}^{N\times N}$ and $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N] \in \mathbf{R}^{N\times N}$ have orthonormal columns, i.e. $U^TU = V^TV = I$, and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_N)$, where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_N \geq 0$ are the singular values of $A$. We have assumed that $\sigma_N > 0$. As $i$ increases, in the problems we are handling the $\sigma_i$ gradually decay to zero and settle to values of the same magnitude of the machine precision. Hence the condition number of $A$, given by $\kappa = \sigma_1/\sigma_N$ is very large. At the same time the $\boldsymbol{u}_i$, $\boldsymbol{v}_i$ become more and more oscillatory, i.e. tend to have more sign changes in their elements. From

$$A = \sum_{i=1}^{N} \sigma_i \boldsymbol{u}_i\, \boldsymbol{v}_i^T \quad \text{and} \quad A^{-1} = \sum_{i=1}^{N} \frac{1}{\sigma_i}\, \boldsymbol{v}_i\, \boldsymbol{u}_i^T,$$

we get the expansion of $\widetilde{\boldsymbol{x}}$ in the basis $V$

$$\widetilde{\boldsymbol{x}} = A^{-1}\boldsymbol{b} = V\Sigma^{-1}U^T\boldsymbol{b} = \sum_{i=1}^{N} \widetilde{x}_i\, \boldsymbol{v}_i, \quad \text{where} \quad \widetilde{x}_i = \frac{\boldsymbol{u}_i^T\boldsymbol{b}}{\sigma_i}. \tag{30}$$

Because of the increasing sign changes, the vectors $\boldsymbol{v}_i$ are associated to frequencies which increase with $i$. The coefficient $\widetilde{x}_i$ measures the contribution of $\boldsymbol{v}_i$ to $\widetilde{\boldsymbol{x}}$, i.e. the amount of information of the $i$th frequency. Generally the low-frequency components describe the overall features of $\widetilde{\boldsymbol{x}}$ and the high-frequency ones represent its details. Let

$$\boldsymbol{\eta} = \sum_{i=1}^{n} \eta_i\, \boldsymbol{u}_i$$

be the expansion of $\boldsymbol{\eta}$ in the basis $U$. Then

$$\boldsymbol{x}^* = A^{-1}\boldsymbol{b}^* = \sum_{i=1}^{N} x_i^*\, \boldsymbol{v}_i, \quad \text{where} \quad x_i^* = \frac{\boldsymbol{u}_i^T\boldsymbol{b}^*}{\sigma_i},$$

and

$$\begin{aligned} \widetilde{\boldsymbol{x}} &= \boldsymbol{x}^* + A^{-1}\boldsymbol{\eta} = \boldsymbol{x}^* + \sum_{i=1}^{N} \frac{1}{\sigma_i}\, \boldsymbol{v}_i\boldsymbol{u}_i^T \sum_{j=1}^{N} \eta_j \boldsymbol{u}_j = \boldsymbol{x}^* + \sum_{i=1}^{N} \frac{\eta_i}{\sigma_i}\, \boldsymbol{v}_i \\ &= \sum_{i=1}^{N} \widetilde{x}_i\, \boldsymbol{v}_i, \quad \text{where} \quad \widetilde{x}_i = x_i^* + \frac{\eta_i}{\sigma_i}. \end{aligned} \tag{31}$$

The coefficients $\eta_i$ are typically of the same order for all $i$. We assume that they are unbiased and uncorrelated with zero mean and variance $\sigma^2$. If $A$ has singular values $\sigma_i$ much smaller than the magnitude of $\boldsymbol{\eta}$, the quantities $\eta_i/\sigma_i$ greatly increase with $i$. It follows that the low-frequency components of $\widetilde{\boldsymbol{x}}$ and $\boldsymbol{x}^*$ do not differ much, while the high-frequency components of $\widetilde{\boldsymbol{x}}$ are disastrously dominated by the high-frequency components of the noise and $\widetilde{\boldsymbol{x}}$ can be affected by a large error with respect to $\boldsymbol{x}^*$. Hence the contribution of the high-frequency components of the noise should be damped. This can be accomplished by using filtering methods which reconstruct solutions of the form

$$\boldsymbol{x}_{\mathrm{reg}} = \sum_{i=1}^{N} \varphi_i\, \widetilde{x}_i\, \boldsymbol{v}_i, \tag{32}$$

where the coefficients $\varphi_i$, called *filter factors*, are close to 1 for small $i$ (say $i \leq \tau$) and much smaller than 1 for large $i$ ($\gg \tau$). This threshold $\tau$ depends on the (in general not known) singular values of $A$ and on the magnitude of the noise (which is assumed to be known in its general features). The determination of a suitable $\tau$ is not an easy task, because a too small $\tau$ results in an oversmoothed reconstruction, where the noise is filtered out together with precious

high-frequency components of $\boldsymbol{x}^*$. On the contrary, a too large $\tau$ results in an undersmoothed reconstruction, where the high-frequency components of the noise contaminate the computed solution. An acceptable approximation can be obtained only if the quantities $|\boldsymbol{u}_i^T \boldsymbol{b}^*|$ decay to zero with $i$ faster than the corresponding $\sigma_i$ (this condition is known as Picard discrete condition), at least until the numerical levelling of the singular values.

Any iterative method which enjoys the semi-convergence property, i.e. in presence of the noise it reconstructs first the low-frequency components, can be employed as a regularizing method. The regularizing properties of CG applied to $A$ in the SPD case or to normal equations in the non SPD case, are well known [25].

## 5.1 Regularization with CG

Let $\boldsymbol{r}_0$ be the initial residual and $W_n$ the orthonormal matrix obtained by applying Lanczos algorithm to vector $\boldsymbol{v} = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$. The columns of $W_n$ form an orthonormal basis of the Krylov subspace $\mathcal{K}_n = \mathcal{K}_n(A, \boldsymbol{r}_0)$. The matrix

$$T_n = W_n^T \, A \, W_n$$

can be viewed as a representation of $A$ projected onto $\mathcal{K}_n$. We already know that the residual $\boldsymbol{r}_n = A(\boldsymbol{x}^* - \boldsymbol{x}_n)$ is orthogonal to $\mathcal{K}_n$. We have seen in Section 3.1 that $\boldsymbol{r}_n = \alpha \, \pi_n(A)\boldsymbol{r}_0$, where $\boldsymbol{p}_n(\lambda) = \det(\lambda I - T_n)$ and $\alpha$ is a scalar $\neq 0$. From (23) it follows that the polynomial $\xi(\lambda)$ which realizes the minimum is equal to $\pi_n(\lambda)/\pi_n(0)$ where $\pi_n(\lambda)$ is defined in (7). Then

$$\boldsymbol{x}^* - \boldsymbol{x}_n = \rho_n(A)(\boldsymbol{x}^* - \boldsymbol{x}_0), \quad \text{where} \quad \rho_n(\lambda) = \frac{\pi_n(\lambda)}{\pi_n(0)} = \prod_{j=1}^{n} \Big(1 - \frac{\lambda}{\theta_j^{(n)}}\Big). \tag{33}$$

From now on, we assume $\boldsymbol{x}_0 = \boldsymbol{0}$. Then $\boldsymbol{r}_n = A\rho_n(A)\boldsymbol{x}^*$. From (22) we get the following three term recurrence for the polynomials $\rho_j(\lambda)$

$$\rho_{j+1}(\lambda) = (\tau_j - \alpha_j\lambda)\rho_j(\lambda) + (1 - \tau_j)\rho_{j-1}(\lambda), \text{ with } \rho_{-1}(\lambda) = 0, \ \rho_0(\lambda) = 1, \tag{34}$$

where

$$\tau_0 = 1, \quad \tau_j = 1 + \frac{\alpha_j \beta_{j-1}}{\alpha_{j-1}} \quad \text{for} \quad j \geq 1,$$

(the same recurrence can be obtained from (7). Using (34) we can compute any value of $\rho_n(\lambda)$, without computing explicitly the Ritz values of $A$.

From (30)

$$\boldsymbol{x}_n = (I - \rho_n(A)) \, \boldsymbol{x}^* = \sum_{i=1}^{N} \frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} (I - \rho_n(A)) \, \boldsymbol{v}_i,$$

and

$$\rho_n(A) \, \boldsymbol{v}_i = \prod_{j=1}^{n} \Big(I - \frac{A}{\theta_j^{(n)}}\Big) \, \boldsymbol{v}_i = \prod_{j=1}^{n} \Big(I - \frac{\sigma_i}{\theta_j^{(n)}}\Big) \, \boldsymbol{v}_i.$$

Hence

$$\boldsymbol{x}_n = \sum_{i=1}^{N} \varphi_i^{(n)} \frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} \, \boldsymbol{v}_i, \quad \text{where} \quad \varphi_i^{(n)} = 1 - \prod_{j=1}^{n} \Big(1 - \frac{\sigma_i}{\theta_j^{(n)}}\Big) = 1 - \rho_n(\sigma_i). \tag{35}$$

Comparing with (30) we see that these $\varphi_i^{(n)}$ are the filter factors of CG at the $n$th iteration. Thus the computation of the filters factors require the knowledge of the singular values of $A$.

As the iteration progresses, more Ritz values converge to the singular values of $A$, hence more filter factors approach 1. It follows that the rate of convergence of CG greatly depends on the way the Ritz values evolve with respect to the singular values of $A$. What can be said is that the filter factors $\varphi_i^{(n)}$ corresponding to the largest $\sigma_i$ approach 1 first, giving at the beginning

the low-frequency components of the solution. Higher-frequency components are reconstructed afterwards.

CG has in general a good convergence rate and finds quickly an optimal vector $\boldsymbol{x}_{opt}$ which minimizes the error with respect to $\boldsymbol{x}^*$. This behavior can be a disadvantage in the regularization context, because also the high-frequency components enter quickly the computed solution and the error increases sharply after the optimal number $k_{opt}$ of steps. As a matter of fact, the determination of $k_{opt}$ is very sensitive to the perturbation of the right-hand side. As a consequence, the regularizing efficiency of CG depends heavily on the effectiveness of the stopping rule employed. A popular stopping procedure implements the *Generalized Cross Validation* rule (GCV), which has the advantage over other rules, of not requiring information on the noise magnitude. Its application with CG is analyzed in [11].

## 5.2 Statistical test of the regularizing behavior

Less attention has been paid, for what concerns regularization, to other Krylov subspace methods, which on the contrary should be taken into consideration because of the slowness of the CGNR convergence in presence of severe ill-conditioning. The regularizing properties of GMRES have been analyzed in [5] from a theoretical point of view. The regularizing properties of BiCG and QMR methods have been tested in [6] on an experimental basis. In [4] tools have been proposed for measuring the regularizing efficiency and the consistency with the discrepancy principle, which is another widely used stopping technique, of different iterative methods of Krylov type.

From a theoretical point of view, the regularizing efficiency of a method should be measured through some norm of the error between the expected solution and the computed solution. But in practice this error cannot be exactly monitored and depends on the rule implemented to stop the iteration. The discrepancy principle is often suggested as a stopping rule, as long as a realistic estimate of the magnitude of the data noise is available.

In [4] this aspect is analyzed statistically for some commonly used Krylov subspaces methods applied to test problems of two sets: the first ones arise from the discretization of Fredholm integral equations of first kind and are taken from the collection [17], the second ones are stochastic and depend on parameters which control the conditioning of the matrix, the decay of its singular values and the compliance with the Picard condition, following the suggestions given in [18], Section 4.7.

CGNR and GMRES, in agreement with known theoretical results, show a good consistency behavior. The other methods taken into consideration show a lower degree of consistency, performing differently on the two sets of problems. In particular, QMR performs better on the problems of the first set in agreement with the results of [6], while CGS, BiCG and Bi-CGSTAB perform better on the problems of the second set. The worst performances of the last three methods can be attributed both to the behavior of the residual norm, whose monotonical decrease is not guaranteed, and to the large convergence speed, which does not allow enough iterations before the noise contaminates the computed solution. If the use of the discrepancy principle implies an overestimation on the number of iterations to be performed, too many iterations would completely alter the result.

Restricting our attention to CG, the stopping index can be estimated through the minimum of the GCV function, whose denominator requires the computation of the trace of the CG influence matrix. GCV has been shown to be very effective when applied to iterative methods whose influence matrix does not depend on the noise, i.e. when the regularized solution depends linearly on the right-hand side of the system. However, this is not the case of CG, and some techniques have been proposed to overcome this drawback. In order to approximate the denominator of the GCV function, in [11] a method which linearizes the dependence of the regularized solution on the noise as suggested, approximating the required derivatives by finite differences. Our aim is to compare the effectiveness of this method with other ones proposed in the literature through an extensive numerical experimentation both on 1D and on 2D test problems.

The effectiveness of GCV as a stopping rule for regularizing CG has been examined also in [12].

# 6 Applications: infinite linear systems

We consider here the case that the matrix $A$ and the vector $\boldsymbol{b}$ of (1) have not a finite size but are finitely expressed and bounded in a normed linear space $S$. Many different problems are modelled by infinite systems of linear equations, for examples partial differential problems discretized on unbounded domains. Typically the matrix $A$ is sparse and we assume that $A$ has at most $\omega > 0$ nonzero elements in each row and in each column. We assume also that the system has a solution $\boldsymbol{x}^*$, whose components satisfy $\lim_{i\to\infty} x_i^* = 0$ and that $\|\boldsymbol{x}^*\|$ is finite.

The standard approach for dealing with the infinite size is the one-shot scheme: a system $\widehat{A}\,\boldsymbol{y} = \widehat{\boldsymbol{b}}$ is solved, where $\widehat{A}$ is a large-sized leading principal submatrix of $A$ and $\widehat{\boldsymbol{b}}$ is the corresponding right-hand side. The solution $\widehat{\boldsymbol{y}}$ of this system, followed by infinite zeros, is taken as an approximation of $\boldsymbol{x}^*$. The large size and the sparseness of the matrix $\widehat{A}$ suggest the use of an iterative method to compute $\widehat{\boldsymbol{y}}$. Let $\boldsymbol{y}^{(i)}$ be the vector computed at the $i$th iteration. The stopping criterion usually implemented correlates the residual $\|\widehat{\boldsymbol{b}} - \widehat{A}\,\boldsymbol{y}^{(i)}\|$ with a tolerance $\tau$, which should be estimated a priori. The choice of $\tau$ is critical.

To lower the computational cost and to individuate a right value for $\tau$, in [10] an adaptive strategy has been suggested, exploiting an inner-outer iterative scheme. For the outer iterations, an increasing sequence $n_k$ of integers determines a sequence of truncated systems of sizes $n_k$

$$A_k \boldsymbol{y} = \boldsymbol{b}_k, \quad k = 1, 2, \ldots, \tag{36}$$

where $A_k$ is the leading $n_k \times n_k$ principal submatrix of $A$. We assume that $A_k$ is nonsingular for any $k$ and approximate the solutions $\boldsymbol{y}_k^*$ by an iterative inner method. We assume that the sequence of the infinite vectors, obtained by completing with zeros $\boldsymbol{y}_k^*$, converges in norm to $\boldsymbol{x}^*$.

For example, the following hypotheses on $A$ and $\boldsymbol{b}$ guarantee such a convergence and are always satisfied by the stochastic problems:
(a) $A$ is columnwise diagonally dominant, with positive elements on the diagonal and nonpositive elements outside,
(b) the right hand-side $\boldsymbol{b}$ and the solution $\boldsymbol{x}^*$ are nonnegative.

The idea on which this inner-outer scheme relies, is that the first components of $\boldsymbol{x}^*$, which are typically the largest ones, should be approximated first, when the cost of the iteration is lower. At each outer step, the initial vector used for the inner iteration is set to the last value obtained at the previous iteration. This requires that the used iterative method takes advantage of a starting point which is closer to the solution. The efficiency of the whole scheme depends on how much the iterative inner method benefits from a starting point closer to the solution.

In the paper [**?**] the authors use a Krylov subspace method as the inner method. The use of Krylov subspace methods is not trivial, since for nonsymmetric systems they frequently show an irregular convergence behavior. In order to analyze how a Krylov subspace method is influenced by the closeness of the starting point to the solution, a proximity property is introduced to answer the question: does a smaller initial $\|\boldsymbol{r}^{(0)}\|$ always result in a smaller number of iterations to achieve a fixed tolerance?

To answer this question, crucial for the effectiveness of the inner-outer scheme, we consider a linear system $H\boldsymbol{z} = \boldsymbol{c}$ to be solved by an iterative method which starts with an initial point $\boldsymbol{z}^{(0)}$ and converges to the solution $\boldsymbol{z}^*$. Let $\boldsymbol{r}^{(0)} = \boldsymbol{c} - H\boldsymbol{z}^{(0)}$ be the starting residual. Let $m$ be the size of $H$. Let $\boldsymbol{s}$ be a random variable uniformly distributed on $\mathbf{R}^m$ with $\|\boldsymbol{s}\| = 1$. The starting vector has the form $\boldsymbol{z}^{(0)} = \boldsymbol{z}^* + \rho\,\boldsymbol{s}/\|H\boldsymbol{s}\|$, where $\rho$ is a parameter. Then $\|\boldsymbol{r}^{(0)}\| = \rho$. Let $f_\lambda(\boldsymbol{z}_0)$ be the smallest number $i$ of iterations required to obtain $\|\boldsymbol{r}^{(i)}\|/\|\boldsymbol{r}_0\| \le \lambda$, where $\boldsymbol{r}^{(i)}$ is the residual at the $i$th iteration. We say that the method enjoys the *proximity property* for the system $H\boldsymbol{z} = \boldsymbol{c}$ if for any $\lambda < 1$ the mean value $\mu\big(f_\lambda(\boldsymbol{z}_0)\big)$ is independent from the parameter $\rho$.

In other words a method enjoys the proximity property if the mean number of iterations required to achieve a fixed tolerance decreases when $\rho$ decreases. We are interested in investigating whether a Krylov subspace method enjoys the proximity property. For GMRES and QMR methods we expect a positive answer, since for a diagonalizable matrix $H$ the relation $\|\boldsymbol{r}^{(i)}\|_2 / \|\boldsymbol{r}^{(0)}\|_2 \leq q_i(H)$ holds, where $q_i(H)$ is independent of the norm of $\boldsymbol{r}^{(0)}$, but we cannot anticipate anything of other methods.

Using a statistical approach, some selected Krylov subspace methods have been applied to a set of large-sized test problems with different starting points, monitoring the reduction of the residuals. When the method did not appear to converge or a breakdown occurred (this happened in 5% of the cases for GMRES and QMR and in 25% of the cases for the other methods) the observation was discarded. A nonparametric Kruskal-Wallis test confirmed that the considered methods enjoyed the proximity property, although for BiCG, Bi-CGSTAB and CGS there was only a limited set of observations accepted for certain problems.

The convergence of the inner iterative method is monitored through the *local* residual norm $\ell_k^{(i)}$ by the stopping condition

$$\ell_k^{(i)} = \|\boldsymbol{b}_k - A_k\, \boldsymbol{y}_k^{(i)}\| \leq \tau_k, \tag{37}$$

where $\tau_k$ is a suitable tolerance. A strictly decreasing sequence $\tau_k$, chosen without taking into account the characteristics of the problem, might not guarantee the convergence of $\boldsymbol{y}_k^{(i)}$ to $\boldsymbol{y}_k^*$. In fact, a vector $\boldsymbol{y}_k^{(i)}$ satisfying (37) can differ from $\boldsymbol{y}_k^*$ by an error which increases with $k$, since $\|\boldsymbol{y}_k^* - \boldsymbol{y}_k^{(i)}\| \leq \|A_k^{-1}\|\, \ell_k^{(i)}$. Indeed, when dealing with infinite size problems, a decreasing sequence of residual norms can be associated with an increasing sequence of errors. To guarantee that a correct numerical approximation of $\boldsymbol{x}^*$ can be achieved by controlling the residual norm, the sequence $\tau_k$ should verify $\lim_{k\to\infty} \|A_k^{-1}\|\tau_k = 0$.

An efficient practical stopping condition can be devised by exploiting the proximity property, requiring the vector $\boldsymbol{y}_k^*$ to minimize in the least squares sense the initial residual norm for the $(k+1)$th step, i. e.

$$\boldsymbol{y}_{k+1}^{(0)} = \begin{bmatrix} \boldsymbol{y}_k^* \\ \boldsymbol{0} \end{bmatrix} = \underset{\boldsymbol{v}\in\mathbf{R}^m}{\operatorname{argmin}} \left\| \boldsymbol{b}_{k+1} - A_{k+1} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{0} \end{bmatrix} \right\|_2 = \underset{\boldsymbol{v}\in\mathbf{R}^m}{\operatorname{argmin}} \left\| \begin{bmatrix} \boldsymbol{b}_k - A_k\, \boldsymbol{v} \\ \boldsymbol{c}_k - B_k\, \boldsymbol{v} \end{bmatrix} \right\|_2,$$

where $\boldsymbol{c}_k = \boldsymbol{b}(n_k+1 : n_{k+1})$, $B_k = A(n_k+1 : n_{k+1}, 1 : n_k)$ and $m = n_k$. In this way $\boldsymbol{y}_{k+1}^{(0)}$ would be the closest starting point for the $(k+1)$th system (36).

Vector $\boldsymbol{y}_k^*$ solves the linear system

$$A_k^T A_k \boldsymbol{v} = A_k^T \boldsymbol{b}_k + \boldsymbol{\eta}(\boldsymbol{v}), \quad \text{where} \quad \boldsymbol{\eta}(\boldsymbol{v}) = B_k^T(\boldsymbol{c}_k - B_k \boldsymbol{v}).$$

We can interpret $\boldsymbol{\eta}(\boldsymbol{v})$ as a noise vector and solve this system by applying the techniques used when dealing with ill-posed problems. Thus we approximate $\boldsymbol{y}_k^*$ by computing a regularized solution of the system $A_k^T A_k \boldsymbol{v} = A_k^T \boldsymbol{b}_k$ by means of an iterative method which enjoys the semi-convergence property and using the so-called *discrepancy principle* as a stopping criterion, which suggests that the iteration should be terminated as soon as the residual computed at the $i$th iteration satisfies $\|A_k^T(\boldsymbol{b}_k - A_k \boldsymbol{v}^{(i)})\| \sim \eta_i$, where $\eta_i = \|\boldsymbol{\eta}(\boldsymbol{v}^{(i)})\|$. Unlike what happens in classical reconstruction problems, where the norm of the noise remains constant during the iteration, here the quantity $\eta_i$ depends on $\boldsymbol{v}^{(i)}$. A practical stopping criterion can be derived by replacing the vectors $\boldsymbol{v}^{(i)}$ with the $\boldsymbol{y}_k^{(i)}$ computed at the $k$th outer step. If $\eta_i$ is bounded from below, the iteration should be stopped as soon as $\eta_i$ levels off and

$$\|A_k^T(\boldsymbol{b}_k - A_k \boldsymbol{y}_k^{(i)})\| \leq \|\boldsymbol{\eta}(\boldsymbol{y}_k^{(i)})\|.$$

No look-ahead strategy is implemented in order to avoid the breakdowns due to the fact that the Krylov subspace cannot be expanded any further. For GMRES, this type of breakdown

signals that the exact solution has been found. For the other methods the literature suggests restarting when this happens. From this point of view the size increase in our outer scheme allows more flexibility since, whenever a breakdown occurs, an automatic restart is performed by enlarging the size. For comparative purposes only, CGNR is used to solve systems (36).

Numerical experiments have been performed on a set of 41 nonsymmetric test problems including differential, artificial and stochastic problems. The experiments aim at validating the inner-outer scheme. We set $n_k = 2n_{k-1}$, with $n_1 = 625$ and $k = 1, \ldots, 10$. To compare the performances of the algorithms, we consider the following measures:
$d_{\max}$, the number of achievable digits of each problem at the largest considered size $n_{10} = 32 \cdot 10^4$
$e_{\max}$, the number of correct digits of the solution computed by each algorithm at the final size.
$lp = d_{\max} - e_{\max}$, the *precision loss*. A large value for $lp$ indicates that many digits have been lost. It typically occurs when the convergence is very slow or when too many breakdowns occur. We assume that $lp \leq 3$ is acceptable.
$cc$, the *computational cost*, obtained by considering the number of matrix-vector products multiplied by the current size (in the table the quantity $c = \text{round}(10^{-6}cc)$ is shown).

The results are presented in Table 1, which describes the average behavior of the methods (varying the problems). Column # gives the number of successes ($lp \leq 3$). The corresponding averaged quantities $lp_{av}$ and $c_{av}$ are computed by considering only the successes.

| method | # | $lp_{av}$ | $c_{av}$ |
|---|---|---|---|
| CGNR | 24 | 0.75 | 1275 |
| GMRES | 41 | 0.20 | 741 |
| BiCG | 20 | 0.67 | 77 |
| CGS | 23 | 0.64 | 61 |
| Bi-CGSTAB | 19 | 0.64 | 35 |
| QMR | 35 | 0.28 | 365 |

Table 1: *Average behavior of the methods.*

From Table 1, GMRES appears to be the most reliable of the considered methods. It produces good results (i.e. $lp_{av} = 0.20$) in most cases. QMR has a comparable $lp_{av}$, but for a lower number of items. The other methods give fewer good results and have a greater precision loss. This poor behavior seems to be due to the irregular convergence of the residuals and to the frequent occurrences of breakdowns. But, when they produce good results, they have a lower cost. CGNR does not appear to be competitive.

Anyway, the heuristics adopted as a stopping criterion, which is based on the proximity property, is effective for all the methods we used. This result is a further demonstration of the validity of the proximity property for the methods considered.

The behavior of the individual methods highlights that, for the implementations taken into consideration, only GMRES, and of the biconjugate methods only QMR, seem to be robust enough to be applied blindly. Of course the other biconjugate methods, for other kinds of problems and/or implemented with minimal residual smoothing to force a monotonical decrease of the residual norms and special look-ahead strategies to prevent breakdowns, might give good results.

# 7 Applications: the ranking problems

In this section we analyze the performance of some Krylov methods on specific ranking problems such as the PageRank method and the ranking of heterogeneous networks.

## 7.1 The PageRank computation

Link based ranking techniques view the Web as a directed graph (the Web Graph) $G = (V, E)$, where each of the $N$ pages is a node and each hyperlink is an arc. The problem of ranking web pages consists in assigning a rank of importance to every page based only on the link structure of the Web and not on the actual contents of the page. The intuition behind the PageRank algorithm is that a page is "important" if it is pointed by other pages which are in turn "important". This definition suggests an iterative fixed-point computation for assigning a ranking score to each page in the Web. Formally, in the original model [23], the computation of the PageRank vector is equivalent to the computation of $\boldsymbol{z}^T = \boldsymbol{z}^T P$, where $P$ is a row stochastic matrix obtained from the adjacency matrix of the Web graph $G$. See [22] for a more deep treatment on the characteristics of the model.

In [7] it is shown how the eigenproblem can be rewritten as a sparse linear system proving that the PageRank vector $\boldsymbol{z}$, can be obtained by solving a sparse linear system.

Once the problem is transformed into a sparse linear system we can apply numerous iterative solvers comparing them with the Power method commonly used for computing PageRank. The matrix involved is moreover an $M$-matrix hence the convergence of stationary methods is always guaranteed. In [7] the most common methods for linear system solution such as Jacobi, Gauss-Seidel and Reverse Gauss-Seidel and the corresponding block methods are tested. These methods have been combined with reordering techniques for increasing data-locality, sometime reducing also the spectral radius of the iteration matrices, and hence increasing the rate of convergence of these stationary methods. In particular, schemes for reordering the matrix in block triangular form have been investigated and the performance of classical stationary methods has proven to be highly improved by reordering techniques also when applying the classical Power method. In particular, the combination of reordering techniques and iterative (block or scalar) methods for the solution of linear systems has proved to be very effective leading to a gain up to 90% in time and to 60% in terms of Mflops.

In [15] the effectiveness of many iterative methods for linear systems on a parallel architecture have been tested on the PageRank problem, showing also that the convergence is faster than the simple Power method.

Among the various permutation schemas proposed in [7] a reordering based on sorting the nodes for increasing outdegree followed by a BFS visit of the web graph has shown to be very effective for many algorithms. Figure 1 shows the shape obtained rearranging a real web matrix with 24 million nodes according to this permutations. The web matrix in Figure 1 has a lower

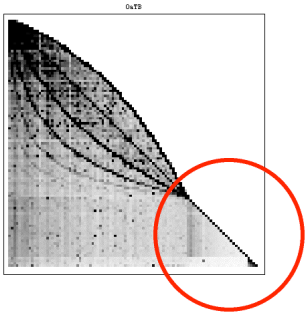


Figure 1: Shape obtained rearranging a Web matrix with 24 million nodes and 100 million links for increasing outdegree and then with the permutation given by a BFS order of visit.

block triangular structure, hence the computation of the PageRank vector can be computed with a forward block substitution. The great advantage in having a block triangular structure is that we can exploit the reducibility of the matrix and use ad hoc methods for different connected

components of the matrix. We investigated this idea, using different iterative procedures for solving the diagonal blocks of the block triangular system. The largest connected component discovered with the BFS order visit of the web graph, is usually huge and for storage constraints we believe that stationary methods are more adequate because they need just one or two vectors for approximating the solution.

The effectiveness of stationary methods is largely discussed in [7] showing that a Reverse Gauss-Seidel technique can be very effective. However, for the tail of the matrix, which is the part circled in Figure 1, we test the effectiveness of different Krylov subspaces iteration methods versus the classical stationary methods such as Jacobi or Gauss-Seidel algorithms. The idea is that we use faster methods on smaller problems where memory space is no more a crucial resource. Note that from previous studies [2] it seems that reordering techniques similar to that considered in this paper can be very effective for improving the convergence of Krylov subspace methods especially when used in combination with incomplete factorization preconditioning techniques.

The experiments described below aim at comparing the most popular Krylov subspaces methods against stationary methods on real matrices resulting from a large web crawl. In order to do that, we consider the tail of the matrix in Figure 1 whose size, once remover the all-zero rows, is something more than 2 Million pages with more than 5 Millions nonzeros (links). From this matrix we extract 1120 matrices whose size ranges from 100 to the full size of the tail.

| Stationary Methods | Krylov Subspace Methods | Preconditioned Methods |
|---|---|---|
| POWER | GMRES(n) | GMRESP(n) |
| JACOBI | BiCG | BiCGP |
| DIRECTGS | BiCGStab | BiCGStabP |
| REVERSEGS | CGS | CGSP |
| | CGNR | CGNRP |
| | QMR | QMRP |

Table 2: List of the stationary and Krylov subspace methods. We considered three restarted version of GMRES with a restart after 10, 20, 40 iteration respectively.

On these matrices we run the methods listed in Figure 2. We consider also preconditioned non stationary methods, which will be denoted by adding a "P" at the end of the name of the method. So, for example, BiCGP stays for preconditioned BiCG method. The use of a preconditioner usually improves the spectral properties of the problem. However, when using a preconditioner one has to evaluate the gain in the speed of convergence versus the cost for constructing it and the increase of the cost of every iteration. We adopt an ILU(0) preconditioner for the non-stationary methods in Figure 2 instead of more complicate and probably more effective preconditioning techniques, because this kind of incomplete factorization is guaranteed to exist for $M$-matrices and the preconditioner can be efficiently stored. In fact, this incomplete factorization produces no others nonzero elements beyond the sparsity structure of the original matrix, hence the preconditioner at works takes exactly as much space to store as the original matrix if the sparse matrix has been stored using a compressed row storage format [1].

Among Generalized Minimal Residual methods we consider only the restarted GMRES. In fact, because of the large size of the problem, we consider only those methods whose storage requirements do not depend on the number of iterations needed to reach convergence. Note that the convergence is guaranteed for the GMRES(n) methods because of the positive definitiveness of the matrices involved, but failures may occur in all the other methods considered. As termination criterion for stationary methods we used a tolerance of $10^{-7}$ on the infinity norm of the difference between two successive iterations. For Krylov methods the same tolerance is tested on the residuals. In general, this stopping criterion may be excessive in the sense that the real error can be much lower than the estimated one, or on the contrary may not guarantee to have a computed solution with 7 digits of precision. In fact, this depends on the particular matrix involved in the system as well as on the starting vector. In order to better evaluate the effectiveness of these

| Methods | $\alpha = 0.5$ | $\alpha = 0.75$ | $\alpha = 0.85$ | $\alpha = 0.95$ | $\alpha = 0.98$ |
|---|---|---|---|---|---|
| POWER | 100 | 100 | 100 | 100 | 100 |
| JACOBI | 100 | 100 | 100 | 100 | 99.7 |
| DIRECTGS | 100 | 100 | 100 | 100 | 99.9 |
| REVERSEGS | 100 | 100 | 100 | 100 | 100 |
| BiCG/BiCGP | 42.3/100 | 10.4/99.5 | 5.9/98 | 5.7/87.6 | 5.6/78 |
| BiCGstab/BiCGstabP | 99.1/100 | 80/100 | 61.2/100 | 37.8/100 | 24.0/100 |
| CGS/CGSP | 59.8/100 | 25.9/100 | 7.0/99.3 | 3.1/92.3 | 2.9/82.1 |
| CGNR/CGNRP | 100/100 | 83.3/99 | 45.0/96 | 23.5/65.3 | 22.5/65.2 |
| QMR/QMRP | 97.4/100 | 93.2/99.9 | 88.9 /99.5 | 72.4/94.4 | 63.6/85.4 |
| GMRES(10)/GMRESP(10) | 100/100 | 100/100 | 100/100 | 94.6/100 | 66.7/100 |
| GMRES(20)/GMRESP(20) | 100/100 | 100/100 | 100/100 | 100/100 | 100/100 |
| GMRES(40)/GMRESP(40) | 100/100 | 100/100 | 100/100 | 100 /100 | 100/100 |

Table 3: Percentage of success of stationary and nonstationary methods over the 1120 matrices whose size ranges from 100 up to 2 Million. The use of the ILU(0) preconditioning technique highly improves the percentage of successes of every method.

methods, the stopping criterion is combined with the control that the infinity norm of the actual error is lower than $10^{-6}$. Hence, a particular trial is considered successful if the actual error is less than $10^{-6}$ even if the termination criteria has not been satisfied. Of course it may happen that the method terminates since the expected tolerance has been reached while the actual error is still greater than $10^{-6}$; in this case a failure is registered. Hence, we may identify three different reasons for failure of a method: failure for breakdown, failure because too many iterations have been performed without meeting the stopping condition or failure because the actual error of the approximated solution is greater than $10^{-6}$ even if the termination conditions were satisfied.

We performed a number of experiments addressing the question of robustness of the iterative method as well as the cost in terms of Mflops counts. The values reported in the tables in the following are obtained averaging over all the 1120 matrices extracted from the tail of the matrix in Figure 1.

A first bunch of experiments has been devoted to study the suitability of the various methods for this particular kind of problems. In particular we measured the percentage of success of every method on the 1120 matrices obtained from the tail of the web matrix depicted in Figure 1. This percentage has been measured for many different values of the parameter $\alpha$ accounting for the probability of the random jump in the random surfer model [22]. From Table 3, we observe that there are methods which are not sufficiently robust even for the customary value of $\alpha = 0.85$. We see that stationary methods and GMRES(n) are very robust, in fact their convergence is also guaranteed theoretically. However, for values of $\alpha$ close to 1 we see that GMRES(10), JACOBI and DIRECTGS methods might fail. This always happens because the methods stopped on the control on the tolerance, while the actual error is still slightly greater than $10^{-6}$. However, GMRES(10) becomes more robust when the ILU(0) preconditioner is used. The reason is that, for $\alpha$ close to 1 the problem is badly conditioned, and the introduction of a preconditioner has the expected behavior of transforming the problem in an equivalent one with more favorable spectral properties. In particular, we see that the use of a preconditioner increases the percentage of successes in all the methods considered; the behavior of BiCGStabP compared with the not preconditioned one is particularly significant. We note that BiCG or CGS methods are not suitable for this kind of problems since they are not sufficiently reliable. Of course, one can try to make more robust these methods implementing them with breakdown free techniques [3] or considering other preconditioning techniques not discussed in this paper.

Since one of the main purposes of this paper is to test which are the methods able to return a good approximation of the solution of the system, we chose to compare only the methods which get a success in all the trials. Table 4 reports the number of iterations and the number of Mflops

| Methods | $\alpha = 0.5$ | $\alpha = 0.75$ | $\alpha = 0.85$ | $\alpha = 0.95$ | $\alpha = 0.98$ |
|---|---|---|---|---|---|
| POWER | 32.05 9.82 | 73.28 23.01 | 125.47 41.03 | 370.97 127.59 | 807.40 247.64 |
| REVERSEGS | 15.80 **2.94** | 34.65 **6.27** | 58.53 **10.67** | 170.92 31.52 | 407.44 75.50 |
| BCGSTABP | **4.00 5.27** | **7.08** 9.25 | **9.67** 13.24 | **16.64 23.79** | **24.68 37.13** |
| GMRES10P | 8.80 7.08 | 14.50 12.43 | 19.47 17.76 | 34.57 36.51 | 54.82 66.88 |
| GMRES20 | 21.04 15.67 | 35.88 33.57 | 48.76 53.28 | 90.06 143.13 | 155.74 336.16 |
| GMRES20P | 8.79 7.37 | 14.02 15.67 | 18.46 21.85 | 30.73 43.07 | 45.83 73.94 |
| GMRES40 | 20.77 21.59 | 33.97 50.15 | 44.54 81.70 | 72.15 205.90 | 105.58 465.92 |
| GMRES40P | 8.79 7.37 | 14.01 16.85 | 18.20 28.68 | 28.89 57.82 | 41.01 94.77 |

Table 4: For every method and for different values of $\alpha$ we report the mean number of iterations and the mean number of Mflops needed to meet the stopping criterion of $10^{-7}$. These values have been obtained averaging over the 1120 matrices of different size. For every value of $\alpha$ in boldface we highlight the best value in terms of average number of iterations and Mflops

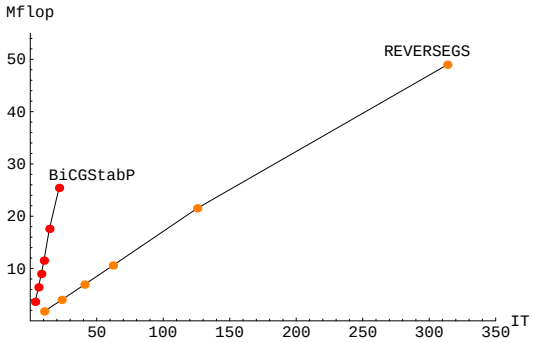



Figure 2: Performance comparison between preconditioner BiCGStab and Reverse Gauss-Seidel methods with the ideal stopping criterion on the actual error of $10^{-6}$. BiCGStabP becomes competitive only for $\alpha \geq 0.9$.

of every "successful" method averaged over all the test matrices. Although the cost of a single iteration changes for every method, we see that there is a big difference in the average number of iterations of stationary and non stationary methods and that among Krylov subspace methods the use of a preconditioner can make the difference. Moreover the number of iteration necessary to reach convergence increases as $\alpha$ goes to 1, meaning that for high values of the damping factor the problem becomes more difficult.

The Mflops count reported as second value in Table 4 is more interesting because it represents the actual cost of the method. Note that for preconditioned methods the operation count includes the time spent in building the preconditioner as well as the extra time required by every iteration. We note that although the iteration count of non stationary methods is lower than that of stationary methods, when one considers the actual cost expressed in Mflops, methods such as REVERSEGS are more convenient than most of the Krylov subspaces methods. In particular, for the usual value of $\alpha = 0.85$ REVERSEGS is on the average better than the other methods. However, for large values of $\alpha$, there is a certain gain in using preconditioned BiCGStab, which requires almost the 50% less than the Reverse Gauss-Seidel method when $\alpha = 0.98$. Moreover, also GMRESP(n) methods become more competitive than REVERSEGS for large values of the jumping probability $\alpha$. Note, that these results should be compared against those obtained using the Power method: in this respect all the successful preconditioned methods behave better for $\alpha > 0.75$. In the ideal situation where one can directly estimate the actual error instead of imposing a stopping criterion on the norm of the residual, we have (see Figure 2) that BiCGStabP is on the average better than REVERSEGS only for large value of $\alpha$.

In Figure 3 the performance in terms of the actual error decay of the two best methods, namely REVERSEGS and the Preconditioned BiCGStab are compared for the largest trail (a 2

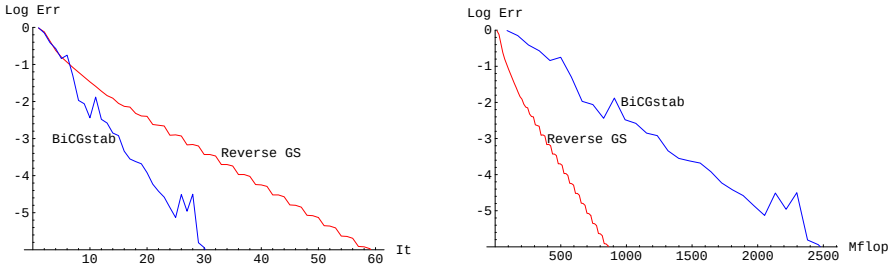



Figure 3: Comparison between the best two methods on the largest trail with $\alpha = 0.85$. In the first graphic it is plotted the behavior of the error as a function of the number of iterations. In the second graphic the error is plotted as a function of the Mflops.

Million size matrix) for $\alpha = 0.85$. In the first figure we show the decay of the error as a function of the number of iterations. We note that BiCGStab does not show a monotony decay while as theoretically predicted we have a linear decay for REVERSEGS. In the second part of Figure 3 the error is plotted as a function of the number of operations necessary expressed in Mflops. We see that computationally REVERSEGS is more convenient than Krylov stationary methods for large matrices and values of $\alpha$ not too close to 1.

## 7.2 Ranking of heterogeneous networks

An heterogeneous network is described by a directed graph $G = (V, E)$ and two mapping functions, one for the nodes $\tau : V \to \mathcal{A}$ and one for the edges $\phi : E \to \mathcal{R}$. Each node $v \in V$ belongs to a particular type $\tau(v) \in \mathcal{A}$ and each edge $e \in E$ belongs to a particular type of relation $\phi(e) \in \mathcal{R}$. Functions $\phi$ and $\tau$ are such that if $e_1$ and $e_2$ are two edges, $e_1 = (v_1, v_2)$ and $e_2 = (w_1, w_2)$, with $\phi(e_1) = \phi(e_2)$, then $\tau(v_1) = \tau(w_1)$ and $\tau(v_2) = \tau(w_2)$. When $|\mathcal{A}| > 1$ and $|\mathcal{R}| > 1$ we say that the graph is a multigraph.

A typical heterogeneous graph is a patent network, where each Patent has associated different features in the set $\mathcal{A} = \{$Patent, Technology, Firm, Examiner, Inventor and Lawyer$\}$. The different relation types are the edges between patents and firms, patents and examiners, patents and the set of inventors, and between patents and lawyers, beside to the edges to other cited patents: each kind of edge has a different semantic meaning, for example the connection between inventors and patents expresses intellectual property over the patent while the edge between patent and examiner represent the fact that a patent was granted by a particular examiner. In Figure 4 it is shown the relations between the different features, and the different kinds of nodes.

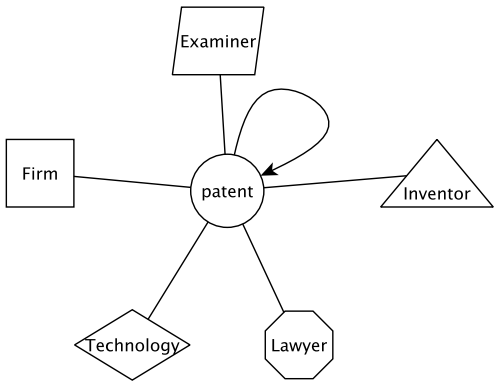



Figure 4: Schema of a patent graph. Each patent has a relation with other types of nodes.

To better understand this kind of networks the information contained and expressed by the different types of relations (edges) between nodes, we associate to the graph a model describing the interaction between items, features and the possible interactions between features. In [8] we define several models and compare them on the basis of a ranking function inspired by the PageRank algorithm. In particular, given the $n_C \times n_C$ citation matrix $C$, let $f = |\mathcal{A}| - 1$, we define the feature matrices $F_k$, for $k = 1, 2, \ldots, f$ of size $n_C \times n_k$, such that the $(i, j)$ entry of $F_k$ is different from zero if item $i$ has attribute $j$ for feature $k$. For patent items, for example, we have the "inventorship feature matrix" storing information about the inventors of a patent, that is, entry $(i, j)$ is nonzero if $j$ is an inventor of patent $i$. We can arrange these matrices and combine then to form a larger matrix $\hat{A}$ such as the one describing the co called `Static model`,

$$\hat{A} = \left[\begin{array}{ccccc|c} \alpha_{1,1} F_1^T C F_1 & \alpha_{1,2} F_1^T F_2 & \cdots & \alpha_{1,f} F_1^T F_f & \alpha_{1,f+1} F_1^T & \\ \alpha_{2,1} F_2^T F_1 & \alpha_{2,2} F_1^T C F_1 & \cdots & \alpha_{2,f} F_2^T F_f & \alpha_{2,f+11} F_2^T & \\ \vdots & \ddots & \ddots & \cdots & \vdots & \boldsymbol{e} \\ \alpha_{f,1} F_f^T F_1 & \cdots & \ddots & \alpha_{f,f} F_f^T C F_f & \alpha_{f,f+1} F_f^T & \\ \alpha_{f+1,1} F_1 & \alpha_{f+1,2} F_2 & \cdots & \alpha_{f+1,f} F_f & \alpha_{f+1,f+1} \tilde{C} & \\ \hline \multicolumn{5}{c|}{\boldsymbol{e}^T} & 0 \end{array}\right].$$

We then normalize by row to get the stochastic irreducible matrix $P = \mathrm{diag}(\hat{A}\boldsymbol{e})^{-1} \hat{A}$. In [8], combining different strategies for the choice of the weights parameters with the different possibilities of arranging the matrices $F_k$ in $\hat{A}$, we obtain 15 different models which are proposed and discussed.

In all our models to compute the rank we have to solve an eigenvector problem involving a stochastic irreducible matrix. More precisely, we have to find the left Perron vector $\boldsymbol{x}$ such that $\boldsymbol{x}^T = \boldsymbol{x}^T P$, with $P$ stochastic. In [8] is is shown in detail how we can reduce the problem to the solution of a sparse linear system where we can directly work on the matrices $F_k$ and $C$ without forming, normalizing and storing the whole matrix $\hat{A}$.

To solve the system we investigated the effectiveness of iterative Krylov methods. In particular, we implemented different Krylov methods, and among them we chose the three more performing: BiCGSTAB, CGS, TFQMR. To refine the final result we add a few step of iterative refinement algorithm.

To test the different algorithms we constructed two datasets with real data extracted from the $US$ patent office and we used five features: Firms, Inventors, Technologies, Lawyers and Examiners. In particular, we denote by $F_1$ the patent-technology matrix where entry $(i, j)$ is one if patent $i$ uses technology $j$; by $F_2$ the patent-firm matrix, recording the firm owning the patent, by $F_3$ the patent-inventors matrix which maps patents to inventors, by $F_4$ the patent-lawyers where each patent is matched to the lawyers applying for the patent, and by $F_5$ the matrix where at each patent is associated the examiners from the US Patent Office who approved the patent. The matrix $C$ contains the citations between patents and is almost triangular since each patent can be based only on patents from the past.

DS1: Consists of $n_C = 2\,474\,786$ US patents from 1976-1990. Of these patents we have additional information that can be grouped into 5 major features, namely $n_1 = 472$ `Technologies`, $n_2 = 165\,662$ `Firms`, $n_3 = 965\,878$ `Inventors`, $n_4 = 25\,341$ `Lawyers` and $n_5 = 12\,817$ `Examiners`, giving rise to a matrix $\hat{A}$ of size $n_C + \sum_{i=1}^{5} n_i$ which is approximately of 3.7 millions.

DS2: Consists of $7\,984\,635$ US patents from 1976-2012. The size of the five features are as follows 475 `Technologies`, 633 551 `Firms`, 4 088 585 `Inventors`, 120 668 `Lawyers` and 64 088 `Examiners`, giving rise to a matrix $\hat{A}$ of size approximately of 13 millions.

When using iterative solvers we have always to address the question of numerical stability. The three proposed methods, `BCGStab`, `CGS` and `TFQMR` have been tested on the two datasets with

| models | BCGstab | | CGS | | TFQMR | |
|---|---|---|---|---|---|---|
| | it | $\log_{10}(res)$ | it | $\log_{10}(res)$ | it | $\log_{10}(res)$ |
| Stiff-U | 18 | -10.49 | 100 | -7.71 | 21 | -3.90 |
| Stiff-D | 23 | -11.77 | 100 | -11.20 | 19 | -4.75 |
| Static-U | 35 | -9.03 | 100 | -6.25 | 40 | -7.83 |
| Static-D | 39 | -11.13 | 100 | -7.22 | 37 | -9.60 |
| Static-DD | 35 | -12.33 | 100 | -12.20 | 30 | -11.99 |
| Heap-U | 32 | -9.86 | 100 | -7.44 | 36 | -8.59 |
| Heap-D | 36 | -11.26 | 100 | -7.73 | 38 | -9.74 |
| Heap-DD | 41 | -11.48 | 100 | -9.46 | 33 | -11.51 |
| Heap-H | 36 | -10.83 | 100 | -6.46 | 30 | -7.72 |
| Heap-HH | 24 | -9.85 | 100 | -7.14 | 27 | -8.38 |
| SHeap-U | 32 | -11.56 | 100 | -8.00 | 29 | -9.91 |
| SHeap-D | 32 | -11.72 | 100 | -8.51 | 28 | -9.85 |
| SHeap-DD | 37 | -11.43 | 100 | -11.83 | 28 | -11.97 |
| SHeap-H | 28 | -10.56 | 100 | -8.33 | 25 | -9.98 |
| SHeap-HH | 29 | -11.34 | 100 | -6.75 | 24 | -10.05 |

Table 5: Performance comparison between three Krylov methods on the 15 models on a problem of size 3.7 million.

| models | BCGstab | | TFQMR | |
|---|---|---|---|---|
| | it | $\log_{10}(res)$ | it | $\log_{10}(res)$ |
| Stiff-U | 14 | -10.57 | 19 | -3.83 |
| Stiff-D | 21 | -11.30 | 25 | -3.71 |
| Static-U | 37 | -6.77 | 52 | -3.09 |
| Static-D | 52 | -10.53 | 53 | -7.89 |
| Static-DD | 39 | -11.38 | 43 | -8.70 |
| Heap-U | 36 | -8.87 | 47 | -7.58 |
| Heap-D | 45 | -6.47 | 51 | -6.11 |
| Heap-DD | 41 | -9.40 | 41 | -6.56 |
| Heap-H | 40 | -9.63 | 43 | -7.31 |
| Heap-HH | 35 | -9.49 | 43 | -7.52 |
| SHeap-U | 40 | -9.78 | 38 | -7.48 |
| SHeap-D | 38 | -10.36 | 36 | -8.10 |
| SHeap-DD | 36 | -11.75 | 34 | -9.79 |
| SHeap-H | 31 | -7.90 | 35 | -4.54 |
| SHeap-HH | 35 | -10.63 | 35 | -5.91 |

Table 6: Performance comparison between two Krylov methods applied to the 15 models on the problem of size 13 million.

an error goal of $10^{-11}$ and with maximum number of iterations equal to 100. For the refinement steps of the power method we set `tol` $= 10^{-13}$. Applying to dataset DS1 the three methods to all the models we obtain the results summarized in Table 5, where instead of the actual residuals we report only their base 10 logarithm.

It is evident that `CGS` is inadequate to cope with this kind of problems since after 100 iterations we have still a high residual norm. Moreover `BCGstab` is better then `TFQMR` since it achieves almost always a lower residual norm. For these reasons we restrict our analysis to `BCGStab` and `TFQMR` comparing them on the dataset of size 13M. We obtain the results reported in Table 6.

We note that `BCGstab` is clearly better than `TFQMR`, but sometimes fails to reach an acceptable

| models | DS1 | size=3.7M | DS2 | size =13M |
|---|---|---|---|---|
| | time(sec.) | $\log_{10}(res)$ | time(sec.) | $\log_{10}(res)$ |
| Stiff-U | 237 | -12.095 | 1078 | -11.952 |
| Stiff-D | 179 | -12.762 | 1422 | -12.1884 |
| Static-U | (*)239 | -10.536 | (*)2314 | -9.0309 |
| Static-D | 188 | -11.740 | 2096 | -10.536 |
| Static-DD | 161 | -13.002 | 1688 | -11.7138 |
| Heap-U | 509 | -11.138 | (*)5992 | -9.7579 |
| Heap-D | 467 | -11.740 | (*)7509 | -10.536 |
| Heap-DD | 450 | -12.535 | (*)5849 | -11.6019 |
| Heap-H | 440 | -11.138 | (*)5978 | -9.93399 |
| Heap-HH | (*)403 | -11.439 | (*)4999 | -10.235 |
| SHeap-U | 80 | -11.740 | (*)717 | -11.1381 |
| SHeap-D | 70 | -11.439 | 661 | -11.4391 |
| SHeap-DD | 67 | -13.107 | 604 | -12.3703 |
| SHeap-H | 86 | -12.041 | (*)662 | -10.8371 |
| SHeap-HH | 60 | -11.689 | 595 | -10.536 |

Table 7: Performance of procedure `SystemSolver` on the 15 models on `DS1` and `DS2`. The results labeled with $(*)$ are those where `TFQMR` has been applied since the required precision of $10^{-11}$ on the residual norm was not satisfied after 100 steps of `BCGStab`.

accuracy. Hence a three step algorithm, described in Procedure `SystemSolver` has been devised.

**Procedure** `SystemSolver`
**Input**: Initial guess $\boldsymbol{x}^{(0)}$, `ErrorGoal`, `maxiter`, `tol`
**Apply** `BCGStab` with error goal=`ErrorGoal` and maximum iterations=`maxiter`
**if** `res` $>$ `ErrorGoal`
  **Apply** `TFQMR` with error goal=`ErrorGoal` and maximum iterations=`maxiter`
**endif**
**Apply** `Iterative Refinement` with tolerance `tol`

Applying this procedure, with `ErrorGoal`=$10^{-10}$, `maxiter`=100 and `tol`=$10^{-13}$, on both the datasets we get the results displayed in Table 7.

From Table 7 we observe that the models which are more stable for the two datasets considered are the `Stiff-D`, `Static-DD`, and among the Heap-like models, we have good performance of `Heap-DD`, `SHeap-DD`.

# References


[1] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J.M. 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 Press, Philadelphia PA., 1993. http://www.netlib.org/linalg/html_templates/report.html

[2] M. Benzi, D. B. Szyld, and A. Van Duin (1999) Orderings for incomplete factorization preconditioning of nonsymmetric problems. *SISC*, **20**(5):1652–1670.

[3] C. Brezinski, M. Redivo-Zaglia, and H. Sadok (1999) New look-ahead Lanczos-type algorithms for linear systems. *Numer. Math.*, **83**:53–85.

[4] P. Brianzi, P. Favati, O. Menchi, F. Romani (2006) A framework for studying the regularizing properties of Krylov subspace methods, *Inverse Problems*, **22**, 1007–1021.

[5] D. Calvetti, B. Lewis and L. Reichel (2002) On the regularizing properties of the GMRES method *Numer. Math.* **91**, 605–625.

[6] D. Calvetti, B. Lewis and L. Reichel (2002) Krylov subspaces iterative methods for non-symmetric discrete ill-posed problems in image restoration *Proc. Society of Photo-Optical Instrumentation Engineers* (SPIE) vol. 4474 (Bellingham, WA: The International Society for Optical Engineering) 224–33.

[7] G. M. Del Corso, A. Gullí, and F. Romani (2005) Fast PageRank computation via a sparse linear system. *Internet Mathematics*, **2**(3), 251–273.

[8] G. M. Del Corso and F. Romani (2015) A multi-class approach for ranking graph nodes: models and experiments with incomplete data. arXiv:1504.07766.

[9] J. Dongarra and F. Sullivan. (2000) Guest editor: introduction to the top 10 algorithms *Computing in Science and Engineering*, **2**(1), 22–23.

[10] P. Favati, G. Lotti, O. Menchi, F. Romani (2007) Adaptive solution of infinite linear systems by Krylov subspace methods, *Journal of Computational and Applied Mathematics*, **210**, 191-199.

[11] P. Favati, G. Lotti, O. Menchi, F. Romani (2014) Generalized Cross-Validation applied to Conjugate Gradient for discrete ill-posed problems, *Applied Mathematics and Computation*, **243**, 258–268.

[12] P. Favati, G. Lotti, O. Menchi, F. Romani (2014) An inner-outer regularizing method for ill-posed problems, *Inverse Problems and Imaging*, **8**, 409–420.

[13] R.W. Freund and N.M. Nachtigal (1991) QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, **60**, 315–339.

[14] R.W. Freund, G.H.Golub and N.M. Nachtigal (1992) Iterative solution of linear systems *Acta Numerica*, 57–100.

[15] D. Gleich, L. Zhukov, and P. Berkhin (2004) Fast parallel PageRank: A linear system approach. Technical Report Yahoo!.

[16] M.H. Gutknecht (2005) A brief introduction to Krylov space methods for solving linear systems, Frontiers of Computational Science – Proceedings of the International Symposium on Frontiers of Computational Science (Y. Kaneda, H. Kawamura, and M. Sasai, eds.), 53–62, Springer-Verlag, Berlin Heidelberg.

[17] P.C. Hansen (1994) Regularization tools: a Matlab package for analysis and solution of discrete ill-posed problems *Numerical Algorithms* **6** 1–35.

[18] P.C.Hansen (1998) *Rank-Deficient and Discrete Ill-Posed Problems* (Philadelphia: SIAM Monographs on Mathematical Modeling and Computation)

[19] R. Hestenes and E. Stiefel (1952) Methods of conjugate gradients for solving linear systems *J. Res. Nat. Bureau Standards*, **49**, 409–435.

[20] A.N. Krylov. On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined (in Russian). Izvestija AN SSSR (News of Academy of Sciences of the USSR), Otdel. mat. i estest. nauk, 1931, VII, Nr.4, 491–539, cited by M. Botchev (2002) "A.N. Krylov, a short biography" http://www.staff.science.uu.nl/ vorst102/kryl.html

[21] C. Lanczos (1952) Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bureau Standards*, **49**, 33–53.

[22] A. N. Langville and C. D. Meyer (2005) A survey of eigenvector methods of Web information retrieval. *SIAM Review*, **47**, 135–161.

[23] L. Page, S. Brin, R. Motwani, and T. Winograd (1998) The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford.

[24] C. C. Paige and M. A. Saunders (1975) Solution of sparse indefinite systems of linear equations*SIAM J. Numer. Anal.*, **12**, 617–629.

[25] R. Plato (1990) Optimal algorithms for linear ill-posed problems yield regularization methods *Num. Funct. Anal. and Optimiz.* **11**, 111–118

[26] Y. Saad and M. H. Schultz (1985) Conjugate gradient-like algorithms for solving nonsymmetric linear systems. *Math. Comp.*, **44**, 417–424.

[27] Y. Saad (1996) Iterative methods for sparse linear systems. PWS Publishing Company, Boston.

[28] P. Sonneveld (1989) CGS, a fast Lanczos-type solver for nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.*, **10**, 36–52.

[29] H. A. van der Vorst (1992) Bi-CGSTAB: a fast and smoothly converging variant of BiCG for the solution of nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.*, **13**, 631–644.