4 Krylov Methods for Linear Systems of Equations

= A class of iterative methods (→ section 3.1) for approximate solution of large linear systems of equations $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{K}^{n,n}$.

BUT, we have reliable *direct* methods (Gauss eliminination → Sect. 2.1, LU-factorization → Alg. 2.2.5, QR-factorization → Alg. **??**) that provide an (apart from roundoff errors) exact solution with a *finite* number of elementary operations!

Alas, direct elimination may not be feasible, or may be grossly inefficient, because

- it may be too expensive (e.g. for $\mathbf{A}$ too large, sparse), → (2.2.1),
- inevitable fill-in may exhaust main memory,
- the system matrix may be available only as procedure `y=evalA(x)` $\leftrightarrow \mathbf{y} = \mathbf{Ax}$

4.1 Descent Methods

Focus: Linear system of equations $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n,n}$, $\mathbf{b} \in \mathbb{R}^n$, $n \in \mathbb{N}$ given,

with symmetric positive definite (s.p.d., → Def. 2.6.2) system matrix $\mathbf{A}$

➡ $\mathbf{A}$-inner product $(\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}^T\mathbf{Ay}$ ⇒ "$\mathbf{A}$-geometry"

Definition 4.1.1 (Energy norm).
A s.p.d. matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ induces an energy norm

$$\|\mathbf{x}\|_A := (\mathbf{x}^T\mathbf{Ax})^{1/2} , \quad \mathbf{x} \in \mathbb{R}^n .$$

Remark 4.1.1 (Krylov methods for complex s.p.d. system matrices).

In this chapter, for the sake of simplicity, we restrict ourselves to $\mathbb{K} = \mathbb{R}$.

However, the (conjugate) gradient methods introduced below also work for LSE $\mathbf{Ax} = \mathbf{b}$ with $\mathbf{A} \in \mathbb{C}^{n,n}$, $\mathbf{A} = \mathbf{A}^H$ s.p.d. when ^{T} is replaced with ^{H} (Hermitian transposed). Then, all theoretical statements remain valid unaltered for $\mathbb{K} = \mathbb{C}$.

△

4.1.1 Quadratic minimization context

Lemma 4.1.2 (S.p.d. LSE and quadratic minimization problem).
A LSE with $\mathbf{A} \in \mathbb{R}^{n,n}$ s.p.d. and $\mathbf{b} \in \mathbb{R}^n$ is equivalent to a minimization problem:

$$\mathbf{Ax} = \mathbf{b} \quad \Leftrightarrow \quad \mathbf{x} = \arg\min_{\mathbf{y}\in\mathbb{R}^n} J(\mathbf{y}) , \quad J(\mathbf{y}) := \tfrac{1}{2}\mathbf{y}^T\mathbf{Ay} - \mathbf{b}^T\mathbf{y} . \tag{4.1.1}$$

A quadratic functional

Proof. If $\mathbf{x}^* := \mathbf{A}^{-1}\mathbf{b}$ a straightforward computation using $\mathbf{A} = \mathbf{A}^T$ shows

$$J(\mathbf{x}) - J(\mathbf{x}^*) = \|\mathbf{x} - \mathbf{x}^*\|_A^2 . \tag{4.1.2}$$

Then the assertion follows from the properties of the energy norm.

Example 4.1.2 (Quadratic functional in 2D).

Plot of J from (4.1.1) for $\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$, $\mathbf{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

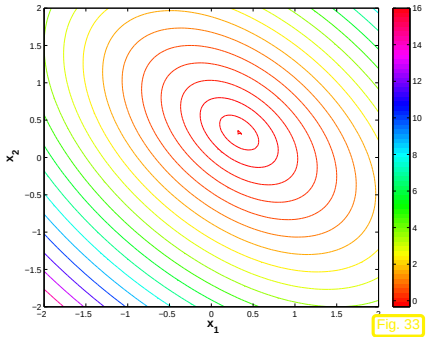


Fig. 33

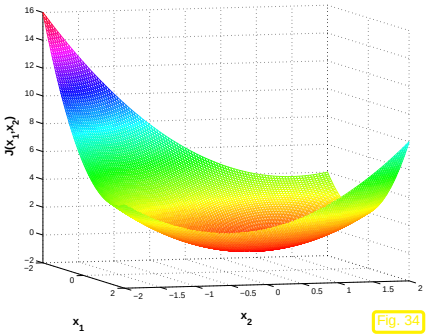


Fig. 34

Level lines of quadratic functionals are (hyper)ellipses

◇

Algorithmic idea: (Lemma 4.1.2 ➢) Solve $\mathbf{Ax} = \mathbf{b}$ iteratively by successive solution of *simpler* minimization problems

4.1.2 Abstract steepest descent

Task:	Given continuously differentiable $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}$, find minimizer $\mathbf{x}^* \in D$: $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in D} F(\mathbf{x})$

Note that a minimizer need not exist, if F is not bounded from below (e.g., $F(x) = x^3$, $x \in \mathbb{R}$, or $F(x) = \log x$, $x > 0$), or if D is open (e.g., $F(x) = \sqrt{x}$, $x > 0$).

The existence of a minimizer is guaranteed if F is bounded from below and D is closed (→ Analysis).

The most natural iteration:

Algorithm 4.1.3 (Steepest descent). (*ger.:* steilster Abstieg)

Initial guess $\mathbf{x}^{(0)} \in D$, $k = 0$
repeat
$\mathbf{d}_k := -\mathbf{grad}\, F(\mathbf{x}^{(k)})$
$t^* := \operatorname{argmin}_{t\in\mathbb{R}} F(\mathbf{x}^{(k)} + t\mathbf{d}_k)$ (line search)
$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + t^*\mathbf{d}_k$
$k := k + 1$
until $\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\| \leq \tau \left\|\mathbf{x}^{(k)}\right\|$

- $\mathbf{d}_k \hat{=}$ *direction of steepest descent*
- linear search $\hat{=}$ 1D minimization: use Newton's method (→ Sect. 3.3.2.1) on derivative
- a posteriori termination criterion, see Sect. 3.1.2 for a discussion. ($\tau \hat{=}$ prescribed tolerance)

The gradient (→ [47, Kapitel 7])

$$\mathbf{grad}\, F(\mathbf{x}) = \begin{pmatrix} \frac{\partial F}{\partial x_i} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{pmatrix} \in \mathbb{R}^n \qquad (4.1.3)$$

provides the direction of local steepest ascent/descent of F

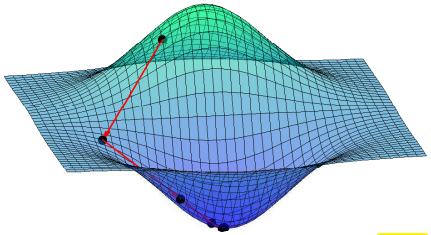


Fig. 35

Of course this very algorithm can encouter plenty of difficulties:

- iteration may get stuck in a *local minimum*,
- iteration may diverge or lead out of D,
- line search may not be feasible.

4.1.3 Gradient method for s.p.d. linear system of equations

However, for the quadratic minimization problem (4.1.1) Alg. 4.1.3 will converge:

Adaptation: steepest descent algorithm Alg. 4.1.3 for quadratic minimization problem (4.1.1)

$$F(\mathbf{x}) := J(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{Ax} - \mathbf{b}^T\mathbf{x} \;\Rightarrow\; \mathbf{grad}\, J(\mathbf{x}) = \mathbf{Ax} - \mathbf{b}\,. \qquad (4.1.4)$$

This follows from $\mathbf{A} = \mathbf{A}^T$, the componentwise expression

$$J(\mathbf{x}) = \tfrac{1}{2}\sum_{i,j=1}^{n} a_{ij}x_ix_j - \sum_{i=1}^{n} b_ix_i$$

and the definition (4.1.3) of the gradient.

➢ For the descent direction in Alg. 4.1.3 applied to the minimization of J from (4.1.1) holds

$$\mathbf{d}_k = \mathbf{b} - \mathbf{Ax}^{(k)} =: \mathbf{r}_k \quad \text{the residual (→ Def. 2.4.5) for } \mathbf{x}^{(k-1)}.$$

Alg. 4.1.3 for $F = J$ from (4.1.1): function to be minimized in line search step:

$$\varphi(t) := J(\mathbf{x}^{(k)} + t\mathbf{d}_k) = J(\mathbf{x}^{(k)}) + t\mathbf{d}_k^T(\mathbf{Ax}^{(k)} - \mathbf{b}) + \tfrac{1}{2}t^2\mathbf{d}_k^T\mathbf{Ad}_k \quad \text{➔ a parabola ! .}$$

$$\frac{d\varphi}{dt}(t^*) = 0 \;\Leftrightarrow\; \boxed{t^* = \frac{\mathbf{d}_k^T\mathbf{d}_k}{\mathbf{d}_k^T\mathbf{Ad}_k}} \quad \text{(unique minimizer) .}$$

Note: $\mathbf{d}_k = 0 \;\Leftrightarrow\; \mathbf{A}\mathbf{x}^{(k)} = \mathbf{b}$ (solution found !)

Note: $\mathbf{A}$ s.p.d. (→ Def. 2.6.2) $\Rightarrow$ $\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k > 0$, if $\mathbf{d}_k \neq 0$

► $\varphi(t)$ is a parabola that is bounded from below (upward opening)

Based on (4.1.4) and (4.1.3) we obtain the following steepest descent method for the minimization problem (4.1.1):

Steepest descent iteration = gradient method for LSE $\mathbf{A}\mathbf{x} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n,n}$ s.p.d., $\mathbf{b} \in \mathbb{R}^n$:

Algorithm 4.1.4 (Gradient method).

Initial guess $\mathbf{x}^{(0)} \in \mathbb{R}^n$, $k = 0$

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$

repeat

$t^* := \dfrac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{r}_k^T\mathbf{A}\mathbf{r}_k}$

$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + t^*\mathbf{r}_k$

$\mathbf{r}_{k+1} := \mathbf{r}_k - t^*\mathbf{A}\mathbf{r}_k$

$k := k + 1$

until $\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\| \leq \tau \left\|\mathbf{x}^{(k)}\right\|$

Code 4.1.6: gradient method for $\mathbf{A}\mathbf{x} = \mathbf{b}$, $\mathbf{A}$ s.p.d.

```
function x = gradit(A,b,x,tol,maxit)
r = b-A*x;
for k=1:maxit
  p = A*r;
  ts = (r'*r)/(r'*p);
  x = x + ts*r;
  if (abs(ts)*norm(r) < tol*norm(x))
    return; end
  r = r - ts*p;
end
```

Recursion for residuals:

$$\mathbf{r}_{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k+1)} = \mathbf{b} - \mathbf{A}(\mathbf{x}^{(k)} + t^*\mathbf{r}_k) = \mathbf{r}_k - t^*\mathbf{A}\mathbf{r}_k\,. \tag{4.1.5}$$

One step of gradient method involves

- *A single* matrix×vector product with $\mathbf{A}$,
- 2 AXPY-operations (→ Sect. **??**) on vectors of length n,
- 2 dot products in $\mathbb{R}^n$.

Computational cost (per step) = cost(matrix×vector) + $O(n)$

➢ If $\mathbf{A} \in \mathbb{R}^{n,n}$ is a sparse matrix (→ Sect. 2.5) with "$O(n)$ nonzero entries", and the data structures allow to perform the matrix×vector product with a computational effort $O(n)$, then a single step of the gradient method costs $O(n)$ elementary operations.

4.1.4 Convergence of gradient method

Example 4.1.7 (Gradient method in 2D).

S.p.d. matrices $\in \mathbb{R}^{2,2}$:

$$\mathbf{A}_1 = \begin{pmatrix} 1.9412 & -0.2353 \\ -0.2353 & 1.0588 \end{pmatrix} \quad , \quad \mathbf{A}_2 = \begin{pmatrix} 7.5353 & -1.8588 \\ -1.8588 & 0.5647 \end{pmatrix}$$

Eigenvalues: $\sigma(\mathbf{A}_1) = \{1, 2\}$, $\sigma(\mathbf{A}_2) = \{0.1, 8\}$

✎ notation: spectrum of a matrix $\in \mathbb{K}^{n,n}$ $\sigma(\mathbf{M}) := \{\lambda \in \mathbb{C}\colon \lambda \text{ is eigenvalue of } \mathbf{M}\}$

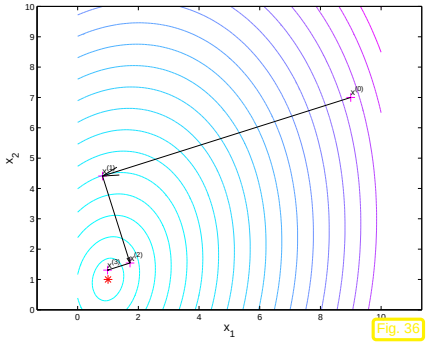


Fig. 36

iterates of Alg. 4.1.4 for $\mathbf{A}_1$

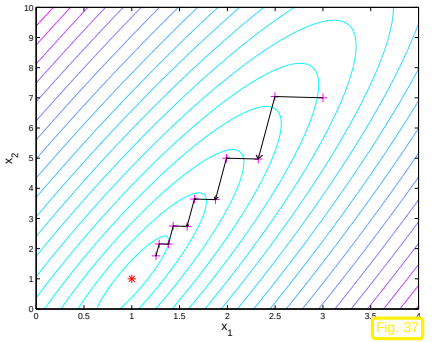


Fig. 37

iterates of Alg. 4.1.4 for $\mathbf{A}_2$

Recall (→ linear algebra) that every real symmetric matrix can be diagonalized by orthogonal similarity transformations, see Cor. 5.1.7: $\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^T$, $\mathbf{D} = \operatorname{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n,n}$ diagonal,

$\mathbf{Q}^{-1} = \mathbf{Q}^T$.

$$J(\mathbf{Q}\widehat{\mathbf{y}}) = \tfrac{1}{2}\widehat{\mathbf{y}}^T\mathbf{D}\widehat{\mathbf{y}} - \underbrace{(\mathbf{Q}^T\mathbf{b})^T}_{=:\widehat{\mathbf{b}}^T}\widehat{\mathbf{y}} = \tfrac{1}{2}\sum_{i=1}^{n} d_i\widehat{y}_i^2 - \widehat{b}_i\widehat{y}_i \ .$$

Hence, a congruence transformation maps the level surfaces of J from (4.1.1) to ellipses with principal axes d_i. As $\mathbf{A}$ s.p.d. $d_i > 0$ is guaranteed.

Observations:

- Larger spread of spectrum leads to slower convergence of gradient method
- Orthogonality of successive residuals $\mathbf{r}_k$, $\mathbf{r}_{k+1}$

Clear from definition of Alg. 4.1.4:

$$\mathbf{r}_k^T\mathbf{r}_{k+1} = \mathbf{r}_k^T\mathbf{r}_k - \mathbf{r}_k^T\frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{r}_k^T\mathbf{A}\mathbf{r}_k}\mathbf{A}\mathbf{r}_k = 0 \ .$$

◇

Example 4.1.8 (Convergence of gradient method).

Convergence of gradient method for diagonal matrices, $\mathbf{x}^* = (1,\ldots,1)^T$, $\mathbf{x}^{(0)} = 0$:

```
d = 1:0.01:2;   A1 = diag(d);
d = 1:0.1:11;   A2 = diag(d);
d = 1:1:101;    A3 = diag(d);
```

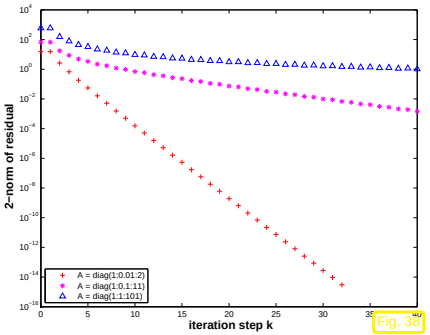


Fig. 39

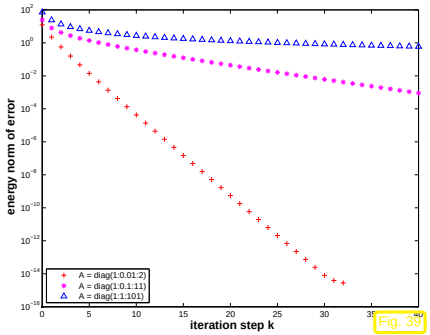


Fig. 39

Note: To study convergence it is *sufficient to consider diagonal matrices*, because

1. for every $\mathbf{A} \in \mathbb{R}^{n,n}$ with $\mathbf{A}^T = \mathbf{A}$ there is an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n,n}$ such that $\mathbf{A} = \mathbf{Q}^T\mathbf{D}\mathbf{Q}$ with a diagonal matrix $\mathbf{D}$ (principal axis transformation, → linear algebra course & Chapter 5, Cor. 5.1.7),
2. when applying the gradient method Alg. 4.1.4 to both $\mathbf{A}\mathbf{x} = \mathbf{b}$ and $\mathbf{D}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}} := \mathbf{Q}\mathbf{b}$, then the iterates $\mathbf{x}^{(k)}$ and $\widetilde{\mathbf{x}}^{(k)}$ are related by $\mathbf{Q}\mathbf{x}^{(k)} = \widetilde{\mathbf{x}}^{(k)}$.

Observation:
- linear convergence (→ Def. 3.1.4), see also Rem. 3.1.3
- rate of convergence increases (↔ speed of convergence decreases) with spread of spectrum of $\mathbf{A}$

Impact of distribution of diagonal entries (↔ eigenvalues) of (diagonal matrix) $\mathbf{A}$
($\mathbf{b} = \mathbf{x}^* = 0$, `x0 = cos((1:n)');`)

Test matrix #1: `A=diag(d); d = (1:100);`
Test matrix #2: `A=diag(d); d = [1+(0:97)/97 , 50 , 100];`
Test matrix #3: `A=diag(d); d = [1+(0:49)*0.05, 100-(0:49)*0.05];`
Test matrix #4: eigenvalues exponentially dense at 1

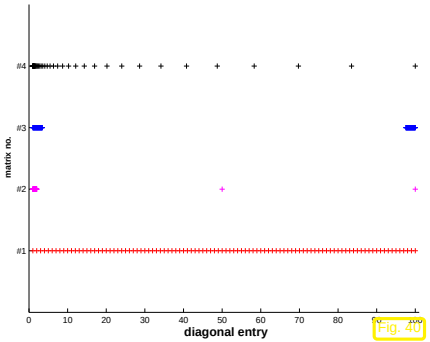
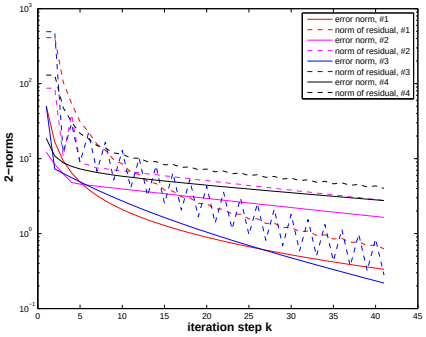


Fig. 40



Observation: Matrices #1, #2 & #4 ➢ little impact of distribution of eigenvalues on *asymptotic* convergence (exception: matrix #2)

◇

Theory [24, Sect. 9.2.2]:

Theorem 4.1.3 (Convergence of gradient method/steepest descent)**.**
The iterates of the gradient method of Alg. 4.1.4 satisfy

$$\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|_A \le L\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\|_A , \quad L := \frac{\operatorname{cond}_2(\mathbf{A})-1}{\operatorname{cond}_2(\mathbf{A})+1} ,$$

that is, the iteration converges at least linearly (w.r.t. energy norm → Def. 4.1.1).

✎ notation: $\operatorname{cond}_2(\mathbf{A})$ ≙ condition number of $\mathbf{A}$ induced by 2-norm

Remark 4.1.9 (2-norm from eigenvalues).

$$\mathbf{A}=\mathbf{A}^T \;\Rightarrow\; \|\mathbf{A}\|_2 = \max(|\sigma(\mathbf{A})|) , \quad \left\|\mathbf{A}^{-1}\right\|_2 = \min(|\sigma(\mathbf{A})|)^{-1} \text{ , if } \mathbf{A} \text{ regular.} \tag{4.1.6}$$

$$\mathbf{A}=\mathbf{A}^T \;\Rightarrow\; \operatorname{cond}_2(\mathbf{A}) = \frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})} , \text{ where } \lambda_{\max}(\mathbf{A}) := \max(|\sigma(\mathbf{A})|) , \; \lambda_{\min}(\mathbf{A}) := \min(|\sigma(\mathbf{A})|) . \tag{4.1.7}$$

✎ other notation $\kappa(\mathbf{A}) := \dfrac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}$ ≙ spectral condition number of $\mathbf{A}$

(for general $\mathbf{A}$: $\lambda_{\max}(\mathbf{A})$/$\lambda_{\min}(\mathbf{A})$ largest/smallest eigenvalue *in modulus*)

These results are an immediate consequence of the fact that

$$\forall \mathbf{A}\in\mathbb{R}^{n,n},\; \mathbf{A}^T=\mathbf{A} \;\; \exists \mathbf{U}\in\mathbb{R}^{n,n},\; \mathbf{U}^{-1}=\mathbf{U}^T:\;\; \mathbf{U}^T\mathbf{A}\mathbf{U} \text{ is diagonal.}$$

→ linear algebra course & Chapter 5, Cor. 5.1.7.

Please note that for general regular $\mathbf{M}\in\mathbb{R}^{n,n}$ we *cannot* expect $\operatorname{cond}_2(\mathbf{M})=\kappa(\mathbf{M})$.

△

4.2 Conjugate gradient method (CG)

Again we consider a linear system of equations $\mathbf{A}\mathbf{x}=\mathbf{b}$ with s.p.d. (→ Def. 2.6.2) system matrix $\mathbf{A}\in\mathbb{R}^{n,n}$ and given $\mathbf{b}\in\mathbb{R}^n$.

Liability of gradient method of Sect. 4.1.3: NO MEMORY

1D line search in Alg. 4.1.4 is oblivious of former line searches, which rules out reuse of information gained in previous steps of the iteration. This is a typical drawback of 1-point iterative methods.

Idea: Replace linear search with subspace correction

Given: • initial guess $\mathbf{x}^{(0)}$
• nested subspaces $U_1\subset U_2\subset U_3\subset\cdots\subset U_n=\mathbb{R}^n$, $\dim U_k = k$

$$\mathbf{x}^{(k)} := \operatorname*{argmin}_{\mathbf{x}\in U_k+\mathbf{x}^{(0)}} J(x) , \tag{4.2.1}$$

quadratic functional from (4.1.1)

Note: Once the subspaces U_k and $\mathbf{x}^{(0)}$ are fixed, the iteration (4.2.1) is well defined, because $J_{|U_k+\mathbf{x}^{(0)}}$ always possess a unique minimizer.

Obvious (from Lemma 4.1.2): $\mathbf{x}^{(n)}=\mathbf{x}^*=\mathbf{A}^{-1}\mathbf{b}$

Thanks to (4.1.2), definition (4.2.1) ensures: $\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|_A \le \left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\|_A$

How to find suitable subspaces U_k **?**

Idea: $U_{k+1} \leftarrow U_k +$ " local steepest descent direction"

given by $-\mathbf{grad}\, J(\mathbf{x}^{(k)}) = \mathbf{b}-\mathbf{A}\mathbf{x}^{(k)} = \mathbf{r}_k$ (residual → Def. 2.4.5)

$$U_{k+1} = \operatorname{Span}\{U_k, \mathbf{r}_k\} , \quad \mathbf{x}^{(k)} \text{ from (4.2.1).} \tag{4.2.2}$$

Obvious: $\mathbf{r}_k=0 \;\Rightarrow\; \mathbf{x}^{(k)}=\mathbf{x}^*:=\mathbf{A}^{-1}\mathbf{b}$ done ✔

Lemma 4.2.1 ($\mathbf{r}_k \perp U_k$)**.**
With $\mathbf{x}^{(k)}$ according to (4.2.1)*, U_k from* (4.2.2) *the residual $\mathbf{r}_k := \mathbf{b}-\mathbf{A}\mathbf{x}^{(k)}$ satisfies*

$$\mathbf{r}_k^T\mathbf{u}=0 \quad \forall \mathbf{u}\in U_k \quad (\text{"}\mathbf{r}_k\perp U_k\text{"}).$$

Proof. Consider

$$\psi(t) = J(\mathbf{x}^{(k)}+t\mathbf{u}) , \quad \mathbf{u}\in U_k, \quad t\in\mathbb{R} .$$

By (4.2.1), $t \mapsto \psi(t)$ has a global minimum in $t = 0$, which implies

$$\frac{d\psi}{dt}(0) = \mathbf{grad}\, J(\mathbf{x}^{(k)})^T \mathbf{u} = (\mathbf{A}\mathbf{x}^{(k)} - \mathbf{b})^T \mathbf{u} = 0 \,.$$

Since $\mathbf{u} \in U_k$ was arbitrary, the lemma is proved. □

Corollary 4.2.2. *If $\mathbf{r}_l \neq 0$ for $l = 0, \ldots, k$, $k \leq n$, then $\{\mathbf{r}_0, \ldots, \mathbf{r}_k\}$ is an orthogonal basis of U_k.*

Lemma 4.2.1 also implies that, if $U_0 = \{0\}$, then $\dim U_k = k$ as long as $\mathbf{x}^{(k)} \neq \mathbf{x}^*$, that is, before we have converged to the exact solution.

(4.2.1) and (4.2.2) define the conjugate gradient method (CG) for the iterative solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$

(hailed as a "top ten algorithm" of the 20th century, SIAM News, 33(4))

4.2.1 Krylov spaces

Definition 4.2.3 (Krylov space)**.**
For $\mathbf{A} \in \mathbb{R}^{n,n}$, $\mathbf{z} \in \mathbb{R}^n$, $\mathbf{z} \neq 0$, the l-th Krylov space is defined as

$$\mathcal{K}_l(\mathbf{A}, \mathbf{z}) := \mathrm{Span}\left\{\mathbf{z}, \mathbf{A}\mathbf{z}, \ldots, \mathbf{A}^{l-1}\mathbf{z}\right\} \,.$$

Equivalent definition: $\mathcal{K}_l(\mathbf{A}, \mathbf{z}) = \{p(\mathbf{A})\mathbf{z}$: p polynomial of degree $\leq l\}$

Lemma 4.2.4. *The subspaces $U_k \subset \mathbb{R}^n$, $k \geq 1$, defined by* (4.2.1) *and* (4.2.2) *satisfy*

$$U_k = \mathrm{Span}\left\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\right\} = \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) \,,$$

where $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ is the initial residual.

Proof. (by induction) Obviously $\mathbf{A}\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) \subset \mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0)$. In addition

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}(\mathbf{x}^{(0)} + \mathbf{z}) \quad \text{for some } \mathbf{z} \in U_k \quad \Rightarrow \quad \mathbf{r}_k = \underbrace{\mathbf{r}_0}_{\in \mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0)} + \underbrace{\mathbf{A}\mathbf{z}}_{\in \mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0)} \,.$$

Since $U_{k+1} = \mathrm{Span}\{U_k, \mathbf{r}_k\}$, we obtain $U_{k+1} \subset \mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0)$. Dimensional considerations based on Lemma 4.2.1 finish the proof. □

4.2.2 Implementation of CG

Assume: basis $\{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$, $l = 1, \ldots, n$, of $\mathcal{K}_l(\mathbf{A}, \mathbf{r})$ available

(4.2.1) ➢ set $\mathbf{x}^{(l)} = \mathbf{x}^{(0)} + \gamma_1 \mathbf{p}_1 + \cdots + \gamma_l \mathbf{p}_k$.

For $\psi(\gamma_1, \ldots, \gamma_l) := J(\mathbf{x}^{(0)} + \gamma_1 \mathbf{p}_1 + \cdots + \gamma_l \mathbf{p}_l)$ holds

$$(4.2.1) \quad \Leftrightarrow \quad \frac{\partial \psi}{\partial \gamma_j} = 0 \,, \quad j = 1, \ldots, l \,.$$

This leads to a linear system of equations by which the coefficients γ_j can be computed:

$$\begin{pmatrix} \mathbf{p}_1^T \mathbf{A}\mathbf{p}_1 & \cdots & \mathbf{p}_1^T \mathbf{A}\mathbf{p}_l \\ \vdots & & \vdots \\ \mathbf{p}_l^T \mathbf{A}\mathbf{p}_1 & \cdots & \mathbf{p}_l^T \mathbf{A}\mathbf{p}_l \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_l \end{pmatrix} = \begin{pmatrix} \mathbf{p}_1^T \mathbf{r} \\ \vdots \\ \mathbf{p}_l^T \mathbf{r} \end{pmatrix} \,. \tag{4.2.3}$$

Great simplification, if $\{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$ $\mathbf{A}$-orthogonal basis of $\mathcal{K}_l(\mathbf{A}, \mathbf{r})$: $\mathbf{p}_j^T \mathbf{A}\mathbf{p}_i = 0$ for $i \neq j$.

Assume: $\mathbf{A}$-orthogonal basis $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ of $\mathbb{R}^n$ available, such that

$$\mathrm{Span}\{\mathbf{p}_1, \ldots, \mathbf{p}_l\} = \mathcal{K}_l(\mathbf{A}, \mathbf{r}) \,.$$

➢ (Efficient) successive computation of $\mathbf{x}^{(l)}$ becomes possible
(LSE (4.2.3) becomes diagonal !)

Input: : initial guess $\mathbf{x}^{(0)} \in \mathbb{R}^n$
Given: : $\mathbf{A}$-orthogonal bases $\{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$ of $\mathcal{K}_l(\mathbf{A}, \mathbf{r}_0)$, $l = 1, \ldots, n$
Output: : approximate solution $\mathbf{x}^{(l)} \in \mathbb{R}^n$ of $\mathbf{A}\mathbf{x} = \mathbf{b}$

$$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)};$$
$$\textbf{for } j = 1 \textbf{ to } l \textbf{ do } \{ \quad \mathbf{x}^{(j)} := \mathbf{x}^{(j-1)} + \frac{\mathbf{p}_j^T \mathbf{r}_0}{\mathbf{p}_j^T \mathbf{A}\mathbf{p}_j} \mathbf{p}_j \quad \} \tag{4.2.4}$$

Task: Efficient computation of $\mathbf{A}$-orthogonal vectors $\{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$ spanning $\mathcal{K}_l(\mathbf{A}, \mathbf{r}_0)$ *during the iteration*.

Lemma 4.2.1 implies orthogonality $\mathbf{p}_j \perp \mathbf{r}_m := \mathbf{b} - \mathbf{A}\mathbf{x}^{(m)}$, $1 \le j \le m \le l$

$$\mathbf{p}_j^H(\mathbf{b} - \mathbf{A}\mathbf{x}^{(m)}) = \mathbf{p}_j^T\left(\underbrace{\mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}}_{=\mathbf{r}_0} - \sum_{k=1}^{m} \frac{\mathbf{p}_k^T\mathbf{r}_0}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}\mathbf{A}\mathbf{p}_k\right) = 0\ . \tag{4.2.5}$$

(4.2.5) ⇒ Idea: Gram-Schmidt orthogonalization, of residuals $\mathbf{r}_j := \mathbf{b} - \mathbf{A}\mathbf{x}^{(j)}$ w.r.t. $\mathbf{A}$-inner product:

$$\mathbf{p}_1 := \mathbf{r}_0\ ,\ \mathbf{p}_{j+1} := \underbrace{(\mathbf{b} - \mathbf{A}\mathbf{x}^{(j)})}_{=:\mathbf{r}_j} - \sum_{k=1}^{j} \frac{\mathbf{p}_k^T\mathbf{A}\mathbf{r}_j}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}\mathbf{p}_k\ ,\quad j = 1,\ldots,l-1\quad . \tag{4.2.6}$$

Lemma 4.2.5 (Bases for Krylov spaces in CG).
If they do not vanish, the vectors $\mathbf{p}_j$, $1 \le j \le l$, *and* $\mathbf{r}_j := \mathbf{b} - \mathbf{A}\mathbf{x}^{(j)}$, $0 \le j \le l$, *from* (4.2.4), (4.2.6) *satisfy*

(i) $\{\mathbf{p}_1,\ldots,\mathbf{p}_j\}$ *is* $\mathbf{A}$*-orthogonal basis von* $\mathcal{K}_j(\mathbf{A},\mathbf{r}_0)$,

(ii) $\{\mathbf{r}_0,\ldots,\mathbf{r}_{j-1}\}$ *is orthogonal basis of* $\mathcal{K}_j(\mathbf{A},\mathbf{r}_0)$, *cf. Cor. 4.2.2*

Proof. $\mathbf{A}$-orthogonality of $\mathbf{p}_j$ by construction, study (4.2.6).

$$\text{(4.2.4) \& (4.2.6)} \quad\Rightarrow\quad \mathbf{p}_{j+1} = \mathbf{r}_0 - \sum_{k=1}^{j}\frac{\mathbf{p}_k^T\mathbf{r}_0}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}\mathbf{A}\mathbf{p}_k - \sum_{k=1}^{j}\frac{\mathbf{p}_k^T\mathbf{A}\mathbf{r}_j}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}\mathbf{p}_k\ .$$

$$\Rightarrow\quad \mathbf{p}_{j+1} \in \operatorname{Span}\{\mathbf{r}_0,\mathbf{p}_1,\ldots,\mathbf{p}_j,\mathbf{A}\mathbf{p}_1,\ldots,\mathbf{A}\mathbf{p}_j\}\ .$$

A simple induction argument confirms (i)

$$\text{(4.2.6)} \quad\Rightarrow\quad \mathbf{r}_j \in \operatorname{Span}\{\mathbf{p}_1,\ldots,\mathbf{p}_{j+1}\} \quad\&\quad \mathbf{p}_j \in \operatorname{Span}\{\mathbf{r}_0,\ldots,\mathbf{r}_{j-1}\}\ . \tag{4.2.7}$$

$$\blacktriangleright\quad \boxed{\operatorname{Span}\{\mathbf{p}_1,\ldots,\mathbf{p}_j\} = \operatorname{Span}\{\mathbf{r}_0,\ldots,\mathbf{r}_{j-1}\} = \mathcal{K}_l(\mathbf{A},\mathbf{r}_0)}\ . \tag{4.2.8}$$

$$\text{(4.2.5)} \quad\Rightarrow\quad \mathbf{r}_j \perp \operatorname{Span}\{\mathbf{p}_1,\ldots,\mathbf{p}_j\} = \operatorname{Span}\{\mathbf{r}_0,\ldots,\mathbf{r}_{j-1}\}\ . \tag{4.2.9}$$

□

Orthogonalities from Lemma 4.2.5 ➤ short recursions for $\mathbf{p}_k$, $\mathbf{r}_k$, $\mathbf{x}^{(k)}$!

(4.2.5) ⇒ (4.2.6) collapses to $\mathbf{p}_{j+1} := \mathbf{r}_j - \dfrac{\mathbf{p}_j^T\mathbf{A}\mathbf{r}_j}{\mathbf{p}_j^T\mathbf{A}\mathbf{p}_j}\mathbf{p}_j\ ,\quad j = 1,\ldots,l\ .$

recursion for residuals:

(4.2.4) $\blacktriangleright$ $\mathbf{r}_j = \mathbf{r}_{j-1} - \dfrac{\mathbf{p}_j^T\mathbf{r}_0}{\mathbf{p}_j^T\mathbf{A}\mathbf{p}_j}\mathbf{A}\mathbf{p}_j\ .$

Lemma 4.2.5, (i) $\blacktriangleright$ $\mathbf{r}_{j-1}^H\mathbf{p}_j = \left(\mathbf{r}_0 + \sum_{k=1}^{m-1}\dfrac{\mathbf{r}_0^H\mathbf{p}_k}{\mathbf{p}_k^H\mathbf{A}\mathbf{p}_k}\mathbf{A}\mathbf{p}_k\right)^H \mathbf{p}_j = \mathbf{r}_0^H\mathbf{p}_j\ .$

Algorithm 4.2.1 (CG method for solving $\mathbf{A}\mathbf{x} = \mathbf{b}$, $\mathbf{A}$ s.p.d.).

Input : initial guess $\mathbf{x}^{(0)} \in \mathbb{R}^n$
Output : approximate solution $\mathbf{x}^{(l)} \in \mathbb{R}^n$

$\mathbf{p}_1 = \mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$;
for $j = 1$ **to** l **do** {

$\mathbf{x}^{(j)} := \mathbf{x}^{(j-1)} + \dfrac{\mathbf{p}_j^T\mathbf{r}_{j-1}}{\mathbf{p}_j^T\mathbf{A}\mathbf{p}_j}\mathbf{p}_j$;

$\mathbf{r}_j = \mathbf{r}_{j-1} - \dfrac{\mathbf{p}_j^T\mathbf{r}_{j-1}}{\mathbf{p}_j^T\mathbf{A}\mathbf{p}_j}\mathbf{A}\mathbf{p}_j$;

$\mathbf{p}_{j+1} = \mathbf{r}_j - \dfrac{(\mathbf{A}\mathbf{p}_j)^T\mathbf{r}_j}{\mathbf{p}_j^T\mathbf{A}\mathbf{p}_j}\mathbf{p}_j$;

}

Input: initial guess $\mathbf{x} \hat{=} \mathbf{x}^{(0)} \in \mathbb{R}^n$
tolerance $\tau > 0$
Output: approximate solution $\mathbf{x} \hat{=} \mathbf{x}^{(l)}$

$\mathbf{p} := \mathbf{r}_0 := \mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}$;
for $j = 1$ **to** $l_{\max}$ **do** {
$\beta := \mathbf{r}^T\mathbf{r}$;
$\mathbf{h} := \mathbf{A}\mathbf{p}$;
$\alpha := \dfrac{\beta}{\mathbf{p}^T\mathbf{h}}$;
$\mathbf{x} := \mathbf{x} + \alpha\mathbf{p}$;
$\mathbf{r} := \mathbf{r} - \alpha\mathbf{h}$;
if $\|\mathbf{r}\| \le \tau\|\mathbf{r}_0\|$ **then stop**;
$\beta := \dfrac{\mathbf{r}^T\mathbf{r}}{\beta}$;
$\mathbf{p} := \mathbf{r} + \beta\mathbf{p}$;
}

$\blacktriangleright$ 1 matrix×vector product, 3 dot products, 3 AXPY-operations per step:
If $\mathbf{A}$ sparse, $\operatorname{nnz}(\mathbf{A}) \sim n$ ➤ computational effort $O(n)$ per step

Code 4.2.2: basic CG iteration

```
function x = cg(A,b,x,tol,maxit)
r = b - A * x; rho = 1; n0 = norm(r);
for i = 1 : maxit
   rho1 = rho; rho = r' * r;
   if (i == 1), p = r;
   else  beta = rho/rho1; p = r + beta * p; end
   q = A * p; alpha = rho/(p' * q);
   x = x + alpha * p;
   if (norm(b - A * x) <= tol*n0) return; end
   r = r - alpha * q;
end
```

MATLAB-function:

`x=pcg(A,b,tol,maxit,[],[],x0)` : Solve $\mathbf{Ax} = \mathbf{b}$ with at most `maxit` CG steps: stop, when $\|\mathbf{r}_l\| : \|\mathbf{r}_0\| < \mathrm{tol}$.

`x=pcg(Afun,b,tol,maxit,[],[],x0)`: `Afun` = handle to function for computing $\mathbf{A}\times$vector.

`[x,flag,relr,it,resv] = pcg(...)` : diagnostic information about iteration

Remark 4.2.3 (A posteriori termination criterion for plain CG).

For any vector norm and associated matrix norm (→ Def. 2.4.2) hold (with residual $\mathbf{r}_l := \mathbf{b} - \mathbf{A}\mathbf{x}^{(l)}$)

$$\frac{1}{\mathrm{cond}(\mathbf{A})}\frac{\|\mathbf{r}_l\|}{\|\mathbf{r}_0\|} \leq \frac{\left\|\mathbf{x}^{(l)}-\mathbf{x}^*\right\|}{\left\|\mathbf{x}^{(0)}-\mathbf{x}^*\right\|} \leq \mathrm{cond}(\mathbf{A})\,\frac{\|\mathbf{r}_l\|}{\|\mathbf{r}_0\|}\,. \tag{4.2.10}$$

relative decrease of iteration error

(4.2.10) can easily be deduced from the error equation $\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^*) = \mathbf{r}_k$, see Def. 2.4.5 and (2.4.6).

△

4.2.3 Convergence of CG

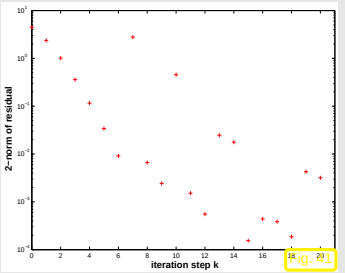
Note: CG is a *direct solver*, because (in exact arithmetic) $\mathbf{x}^{(k)} = \mathbf{x}^*$ for some $k \leq n$

Example 4.2.4 (Impact of roundoff errors on CG).

Numerical experiment: `A=hilb(20)`, $\mathbf{x}^{(0)} = 0$, $\mathbf{b} = (1,\ldots,1)^T$

Hilbert-Matrix: extremely ill-conditioned

residual norms during CG iteration ▷: $\mathbf{R} = \left[\mathbf{r}_0,\ldots,\mathbf{r}^{(10)}\right]$



$\mathbf{R}^T\mathbf{R} =$

$$\begin{pmatrix}
1.000000 & -0.000000 & 0.000000 & -0.000000 & 0.000000 & -0.000000 & 0.016019 & -0.795816 & -0.430569 & 0.348133 \\
-0.000000 & 1.000000 & -0.000000 & 0.000000 & -0.000000 & 0.000000 & -0.012075 & 0.600068 & -0.520610 & 0.420903 \\
0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 & -0.000000 & 0.001582 & -0.078664 & 0.384453 & -0.310577 \\
-0.000000 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 & -0.000024 & 0.001218 & -0.024115 & 0.019394 \\
0.000000 & -0.000000 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 & -0.000002 & 0.000151 & -0.000118 \\
-0.000000 & 0.000000 & -0.000000 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 & -0.000000 & 0.000000 \\
0.016019 & -0.012075 & 0.001582 & -0.000024 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & -0.000000 & 0.000000 \\
-0.795816 & 0.600068 & -0.078664 & 0.001218 & -0.000002 & 0.000000 & -0.000000 & 1.000000 & 0.000000 & -0.000000 \\
-0.430569 & -0.520610 & 0.384453 & -0.024115 & 0.000151 & -0.000000 & -0.000000 & 0.000000 & 1.000000 & 0.000000 \\
0.348133 & 0.420903 & -0.310577 & 0.019394 & -0.000118 & 0.000000 & 0.000000 & -0.000000 & 0.000000 & 1.000000
\end{pmatrix}$$

➢ Roundoff
- destroys orthogonality of residuals
- prevents computation of exact solution after n steps.

Numerical instability (→ Def. **??**) ➢ pointless to (try to) use CG as direct solver!

Practice: CG used for large n as *iterative solver*: $\mathbf{x}^{(k)}$ for some $k \ll n$ is expected to provide good approximation for $\mathbf{x}^*$

Example 4.2.5 (Convergence of CG as iterative solver).

CG (Code 4.2.1)& gradient method (Code 4.1.4) for LSE with sparse s.p.d. "Poisson matrix"

```
A = gallery('poisson',m); x0 = (1:n)'; b = zeros(n,1);
```

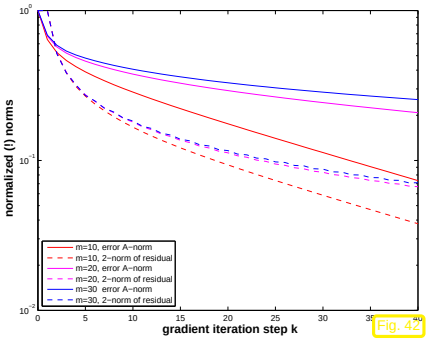


Fig. 42

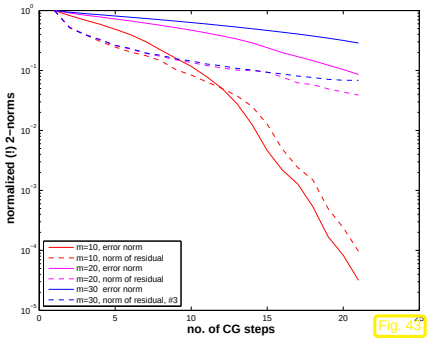


Fig. 43

Observations:

- CG much faster than gradient method (as expected, because it has “memory”)
- Both, CG and gradient method converge more slowly for larger sizes of Poisson matrices.

◇

Convergence theory:

A simple consequence of (4.1.2) and (4.2.1):

Corollary 4.2.6 (“Optimality” of CG iterates)**.**
Writing $\mathbf{x}^* \in \mathbb{R}^n$ for the exact solution of $\mathbf{Ax} = \mathbf{b}$ the CG iterates satisfy

$$\left\|\mathbf{x}-\mathbf{x}^{(l)}\right\|_A = \min\{\|\mathbf{y}-\mathbf{x}\|_A : \mathbf{y} \in \mathbf{x}^{(0)} + \mathcal{K}_l(\mathbf{A},\mathbf{r}_0)\} \quad , \quad \mathbf{r}_0 := \mathbf{b}-\mathbf{Ax}^{(0)} \, .$$

This paves the way for a quantitative convergence estimate:

$$\mathbf{y} \in \mathbf{x}^{(0)} + \mathcal{K}_l(\mathbf{A},\mathbf{r}) \quad \Leftrightarrow \quad \mathbf{y} = \mathbf{x}^{(0)} + \mathbf{A}\,p(\mathbf{A})(\mathbf{x}-\mathbf{x}^{(0)}) \, , \quad p = \text{polynomial of degree } \le l-1 \, .$$

$$\blacktriangleright \quad \mathbf{x}-\mathbf{y} = q(\mathbf{A})(\mathbf{x}-\mathbf{x}^{(0)}), \quad q = \text{polynomial of degree } \le l \, , \quad q(0)=1 \, .$$

$$\left\|\mathbf{x}-\mathbf{x}^{(l)}\right\|_A \le \boxed{\min\{\max_{\lambda\in\sigma(\mathbf{A})} |q(\lambda)| : q \text{ polynomial of degree } \le l \, , \quad q(0)=1\}} \cdot \left\|\mathbf{x}-\mathbf{x}^{(0)}\right\|_A \, . \tag{4.2.11}$$

Bound this minimum for $\lambda \in [\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$ by using suitable “polynomial candidates”

Tool: Chebychev polynomials ➢ lead to the following estimate [24, Satz 9.4.2]

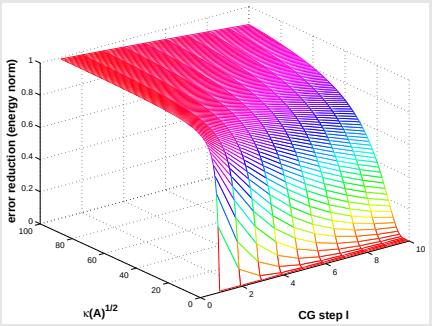
Theorem 4.2.7 (Convergence of CG method)**.**
The iterates of the CG method for solving $\mathbf{Ax} = \mathbf{b}$ (see Code 4.2.1) with $\mathbf{A} = \mathbf{A}^T$ s.p.d. satisfy

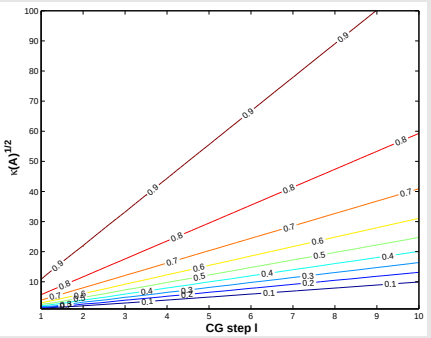
$$\left\|\mathbf{x}-\mathbf{x}^{(l)}\right\|_A \le \frac{2\left(1-\frac{1}{\sqrt{\kappa(\mathbf{A})}}\right)^l}{\left(1+\frac{1}{\sqrt{\kappa(\mathbf{A})}}\right)^{2l}+\left(1-\frac{1}{\sqrt{\kappa(\mathbf{A})}}\right)^{2l}} \le 2\left(\frac{\sqrt{\kappa(\mathbf{A})}-1}{\sqrt{\kappa(\mathbf{A})}+1}\right)^l \left\|\mathbf{x}-\mathbf{x}^{(0)}\right\|_A \, .$$

(recall: $\kappa(\mathbf{A})$ = spectral condition number of $\mathbf{A}$, $\kappa(\mathbf{A}) = \mathrm{cond}_2(\mathbf{A})$)

The estimate of this theorem confirms *asymptotic linear convergence* of the CG method (→ Def. 3.1.4) with a rate of $\dfrac{\sqrt{\kappa(\mathbf{A})}-1}{\sqrt{\kappa(\mathbf{A})}+1}$

Plots of bounds for error reduction (in energy norm) during CG iteration from Thm. 4.2.7:

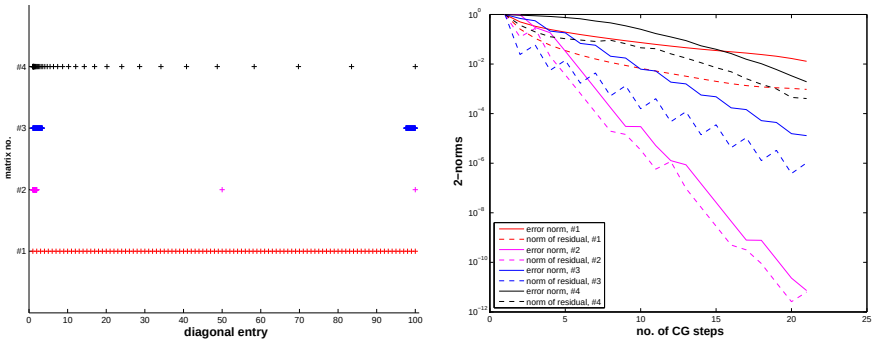




Code 4.2.6: plotting theoretical bounds for CG convergence rate

```
function plottheorate
[X,Y] = meshgrid(1:10,1:100); R = zeros(100,10);
for l=1:100
  t = 1/l;
  for j=1:10
    R(l,j) = 2*(1-t)^j/((1+t)^(2*j)+(1-t)^(2*j));
  end
end

figure; view([-45,28]); mesh(X,Y,R); colormap hsv;
```

```
xlabel('{\bf CG step l}','Fontsize',14);
ylabel('{\bf \kappa(A)^{1/2}}','Fontsize',14);
zlabel('{\bf error reduction (energy norm)}','Fontsize',14);

print -depsc2 '../PICTURES/theorate1.eps';

figure; [C,h] = contour(X,Y,R); clabel(C,h);
xlabel('{\bf CG step l}','Fontsize',14);
ylabel('{\bf \kappa(A)^{1/2}}','Fontsize',14);

print -depsc2 '../PICTURES/theorate2.eps';
```



Observations: Distribution of eigenvalues has crucial impact on convergence of CG
(This is clear from the convergence theory, because detailed information about the spectrum allows a much better choice of "candidate polynomial" in (4.2.11) than merely using Chebychev polynomials)

➢ Clustering of eigenvalues leads to faster convergence of CG
(in stark contrast to the behavior of the gradient method, see Ex. 4.1.8)

CG convergence boosted by clustering of eigenvalues

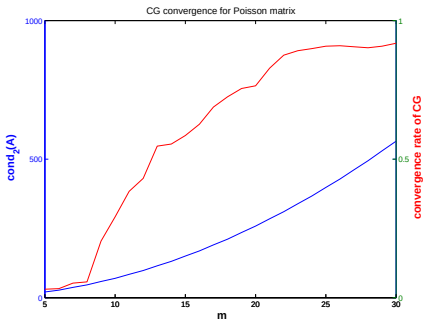
Example 4.2.7 (Convergence rates for CG method).

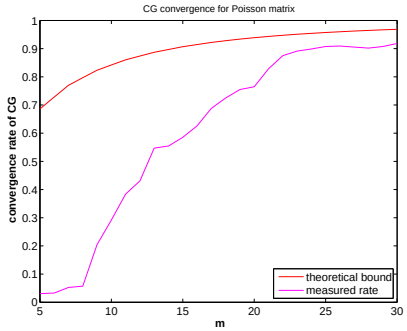
Code 4.2.8: CG for Poisson matrix

```
A = gallery('poisson',m); n = size(A,1);
x0 = (1:n)'; b = ones(n,1); maxit = 30; tol =0;
[x,flag,relres,iter,resvec] = pcg(A,b,tol,maxit,[],[],x0);
```

Measurement of rate of (linear) convergence:

$$\text{rate} \approx \sqrt[10]{\frac{\left\|\mathbf{r}^{(30)}\right\|_2}{\left\|\mathbf{r}^{(20)}\right\|_2}}\,.$$





◇

Example 4.2.9 (CG convergence and spectrum). → Ex. 4.1.8

Test matrix #1: `A=diag(d); d = (1:100);`
Test matrix #2: `A=diag(d); d = [1+(0:97)/97 , 50 , 100];`
Test matrix #3: `A=diag(d); d = [1+(0:49)*0.05, 100-(0:49)*0.05];`
Test matrix #4: eigenvalues exponentially dense at 1

```
x0 = cos((1:n)'); b = zeros(n,1);
```

◇

4.3 Preconditioning

Thm. 4.2.7 ➢ (Potentially) slow convergence of CG in case $\kappa(\mathbf{A}) \gg 1$.

Idea: Preconditioning

Apply CG method to transformed linear system

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}} \quad , \quad \widetilde{\mathbf{A}} := \mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2} \, , \, \widetilde{\mathbf{x}} := \mathbf{B}^{1/2}\mathbf{x} \, , \, \widetilde{\mathbf{b}} := \mathbf{B}^{-1/2}\mathbf{b} \, , \qquad (4.3.1)$$

with "small" $\kappa(\widetilde{\mathbf{A}})$, $\mathbf{B} = \mathbf{B}^T \in \mathbb{R}^{N,N}$ s.p.d. ≙ preconditioner.

What is meant by the "square root" $\mathbf{B}^{1/2}$ of a s.p.d. matrix $\mathbf{B}$?

Recall: for every $\mathbf{B} \in \mathbb{R}^{n,n}$ with $\mathbf{B}^T = \mathbf{B}$ there is an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n.n}$ such that $\mathbf{B} = \mathbf{Q}^T\mathbf{D}\mathbf{Q}$ with a diagonal matrix $\mathbf{D}$ (→ linear algebra, Chapter 5, Cor. 5.1.7). If $\mathbf{B}$ is s.p.d. the

(diagonal) entries of $\mathbf{D}$ are strictly positive and we can define

$$\mathbf{D} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n), \quad \lambda_i > 0 \quad \Rightarrow \quad \mathbf{D}^{1/2} := \operatorname{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n}) .$$

This is generalized to

$$\mathbf{B}^{1/2} := \mathbf{Q}^T \mathbf{D}^{1/2} \mathbf{Q} ,$$

and one easily verifies, using $\mathbf{Q}^T = \mathbf{Q}^{-1}$, that $(\mathbf{B}^{1/2})^2 = \mathbf{B}$ and that $\mathbf{B}^{1/2}$ is s.p.d. In fact, these two requirements already determine $\mathbf{B}^{1/2}$ uniquely.

Notion 4.3.1 (Preconditioner)**.**
*A s.p.d. matrix $\mathbf{B} \in \mathbb{R}^{n,n}$ is called a preconditioner (*ger.: *Vorkonditionierer) for the s.p.d. matrix $\mathbf{A} \in \mathbb{R}^{n,n}$, if*

1. *$\kappa(\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2})$ is "small" and*
2. *the evaluation of $\mathbf{B}^{-1}\mathbf{x}$ is about as expensive (in terms of elementary operations) as the matrix×vector multiplication $\mathbf{A}\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^n$.*

There are several equivalent ways to express that $\kappa(\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2})$ is "small":

- $\kappa(\mathbf{B}^{-1}\mathbf{A})$ is "small",

  because spectra agree $\sigma(\mathbf{B}^{-1}\mathbf{A}) = \sigma(\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2})$ due to similarity (→ Lemma 5.1.4)
- $\exists 0 < \gamma < \Gamma, \quad \Gamma/\gamma$ "'small": $\gamma\,(\mathbf{x}^T\mathbf{B}\mathbf{x}) \le \mathbf{x}^T\mathbf{A}\mathbf{x} \le \Gamma\,(\mathbf{x}^T\mathbf{B}\mathbf{x}) \quad \forall \mathbf{x} \in \mathbb{R}^n$,

  where equivalence is seen by transforming $\mathbf{y} := \mathbf{B}^{-1/2}\mathbf{x}$ and appealing to the min-max Theorem 5.3.5.

"Reader's digest" version of notion 4.3.1:

S.p.d. $\mathbf{B}$ preconditioner $:\Leftrightarrow$ $\mathbf{B}^{-1}$ = cheap approximate inverse of $\mathbf{A}$

Problem: $\mathbf{B}^{1/2}$, which occurs prominently in (4.3.1) is usually not available with acceptable computational costs.

However, if one formally applies Algorithm 4.2.1 to the transformed system from (4.3.1), it becomes apparent that, after suitable transformation of the iteration variables $\mathbf{p}_j$ and $\mathbf{r}_j$, $\mathbf{B}^{1/2}$ and $\mathbf{B}^{-1/2}$ invariably occur in products $\mathbf{B}^{-1/2}\mathbf{B}^{-1/2} = \mathbf{B}^{-1}$ and $\mathbf{B}^{1/2}\mathbf{B}^{-1/2} = \mathbf{I}$. Thus, thanks to this intrinsic transformatin square roots of $\mathbf{B}$ are not required for the implementation!

Algorithm 4.3.1 (Preconditioned CG method (PCG)).

Input: initial guess $\mathbf{x} \in \mathbb{R}^n \hat{=} \mathbf{x}^{(0)} \in \mathbb{R}^n$, tolerance $\tau > 0$
Output: approximate solution $\mathbf{x} \hat{=} \mathbf{x}^{(l)}$

$\mathbf{p} := \mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}$; $\mathbf{p} := \mathbf{B}^{-1}\mathbf{r}$; $\mathbf{q} := \mathbf{p}$; $\tau_0 := \mathbf{p}^T\mathbf{r}$;
for $l = 1$ **to** $l_{\max}$ **do** {
  $\beta := \mathbf{r}^T\mathbf{q}$; $\mathbf{h} := \mathbf{A}\mathbf{p}$; $\alpha := \frac{\beta}{\mathbf{p}^T\mathbf{h}}$;
  $\mathbf{x} := \mathbf{x} + \alpha\mathbf{p}$;
  $\mathbf{r} := \mathbf{r} - \alpha\mathbf{h}$;
  $\mathbf{q} := \mathbf{B}^{-1}\mathbf{r}$; $\beta := \frac{\mathbf{r}^T\mathbf{q}}{\beta}$;
  if $|\mathbf{q}^T\mathbf{r}| \le \tau \cdot \tau_0$ **then stop**;
  $\mathbf{p} := \mathbf{q} + \beta\mathbf{p}$;
  }
(4.3.2)

Computational effort per step: 1 evaluation $\mathbf{A}\times$vector, 1 evaluation $\mathbf{B}^{-1}\times$vector, 3 dot products, 3 AXPY-operations

Remark 4.3.2 (Convergence theory for PCG).

Assertions of Thm. 4.2.7 remain valid with $\kappa(\mathbf{A})$ replaced with $\kappa(\mathbf{B}^{-1}\mathbf{A})$ and energy norm based on $\widetilde{\mathbf{A}}$ instead of $\mathbf{A}$.

△

Example 4.3.3 (Simple preconditioners).

$\mathbf{B}$ = easily invertible "part" of $\mathbf{A}$

- $\mathbf{B} = \texttt{diag}(\mathbf{A})$: Jacobi preconditioner (diagonal scaling)
- $(\mathbf{B})_{ij} = \begin{cases} (\mathbf{A})_{ij} & \text{, if } |i-j| \le k \text{ ,} \\ 0 & \text{else,} \end{cases}$ for some $k \ll n$.
- Symmetric Gauss-Seidel preconditioner

Idea: Solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ approximately in two stages:

① Approximation $\mathbf{A}^{-1} \approx \texttt{tril}(\mathbf{A})$ (lower triangular part): $\widetilde{\mathbf{x}} = \texttt{tril}(\mathbf{A})^{-1}\mathbf{b}$

② Approximation $\mathbf{A}^{-1} \approx \texttt{triu}(\mathbf{A})$ (upper triangular part) and use this to approximately "solve" the error equation $\mathbf{A}(\mathbf{x} - \widetilde{\mathbf{x}}) = \mathbf{r}$, with residual $\mathbf{r} := \mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}$:

$$\mathbf{x} = \widetilde{\mathbf{x}} + \texttt{triu}(\mathbf{A})^{-1}(\mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}) \,.$$

With $\mathbf{L}_A := \texttt{tril}(\mathbf{A})$, $\mathbf{U}_A := \texttt{triu}(\mathbf{A})$ one finds

$$\mathbf{x} = (\mathbf{L}_A^{-1} + \mathbf{U}_A^{-1} - \mathbf{U}_A^{-1}\mathbf{A}\mathbf{L}_A^{-1})\mathbf{b} \quad \blacktriangleright \quad \mathbf{B} = \mathbf{L}_A^{-1} + \mathbf{U}_A^{-1} - \mathbf{U}_A^{-1}\mathbf{A}\mathbf{L}_A^{-1} \,.$$

◇

More complicated preconditioning strategies:

- Incomplete Cholesky factorization, MATLAB-`ichol`
- Sparse approximate inverse preconditioner (SPAI)

Example 4.3.4 (Tridiagonal preconditioning).

Efficacy of preconditioning of sparse LSE with tridiagonal part:

Code 4.3.5: LSE for Ex. 4.3.4

```
A = spdiags(repmat([1/n,-1,2+2/n,-1,1/n],n,1),[-n/2,-1,0,1,n/2],n,n);
b = ones(n,1); x0 = ones(n,1); tol = 1.0E-4; maxit = 1000;
evalA = @(x) A*x;

%no preconditioning
invB = @(x) x; [x,rn] = pcgbase(evalA,b,tol,maxit,invB,x0);

%tridiagonal preconditioning
B = spdiags(spdiags(A,[-1,0,1]),[-1,0,1],n,n);
invB  = @(x) B\x; [x,rnpc] = pcgbase(evalA,b,tol,maxit,invB,x0);
```

Code 4.3.6: simple PCG implementation

```
function [x,rn,xk] = pcgbase(evalA,b,tol,maxit,invB,x)
r = b - evalA(x); rho = 1; rn = [];
if (nargout > 2), xk = x; end
for i = 1 : maxit
  y = invB(r);
  rho_old = rho; rho = r' * y; rn = [rn,rho];
  if (i == 1), p = y; rho0 = rho;
  elseif (rho < rho0*tol), return;
  else beta = rho/rho_old; p = y+beta*p; end
  q = evalA(p); alpha = rho /(p' * q);
  x = x + alpha * p;
  r = r - alpha * q;
  if (nargout > 2), xk = [xk,x]; end
end
```

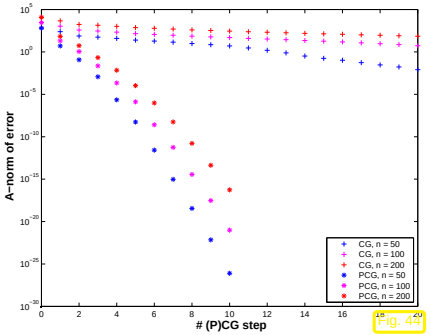



Fig. 44

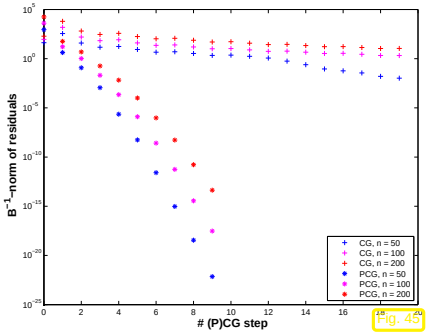


Fig. 45

n	# CG steps	# PCG steps
16	8	3
32	16	3
64	25	4
128	38	4
256	66	4
512	106	4
1024	149	4
2048	211	4
4096	298	3
8192	421	3
16384	595	3
32768	841	3

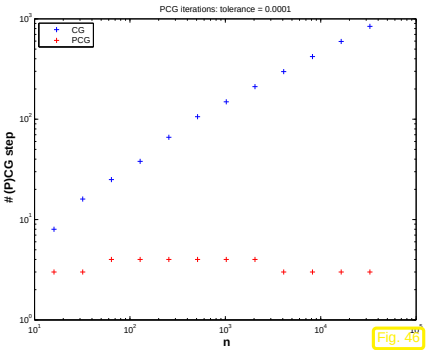


Fig. 46

Clearly in this example the tridiagonal part of the matrix is dominant for large n. In addition, its condition number grows $\sim n^2$ as is revealed by a closer inspection of the spectrum.

Preconditioning with the tridiagonal part manages to suppress this growth of the condition number of $\mathbf{B}^{-1}\mathbf{A}$ and ensures fast convergence of the preconditioned CG method.

◇

Remark 4.3.7 (Termination of PCG).

Rem. 4.2.3, (4.2.10) ➤ Monitor transformed residual

$$\widetilde{\mathbf{r}} = \widetilde{\mathbf{b}} - \widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \mathbf{B}^{-1/2}\mathbf{r} \quad \Rightarrow \quad \|\widetilde{\mathbf{r}}\|_2^2 = \mathbf{r}^T\mathbf{B}^{-1}\mathbf{r} \,.$$

► Estimates for energy norm of error $\mathbf{e}^{(l)} := \mathbf{x} - \mathbf{x}^{(l)}$, $\mathbf{x}^* := \mathbf{A}^{-1}\mathbf{b}$

Use error equation $\mathbf{A}\mathbf{e}^{(l)} = \mathbf{r}_l$:

$$\mathbf{r}_l^T\mathbf{B}^{-1}\mathbf{r}_l = (\mathbf{B}^{-1}\mathbf{A}\mathbf{e}^{(l)})^T\mathbf{A}\mathbf{e}^{(l)} \le \lambda_{\max}(\mathbf{B}^{-1}\mathbf{A})\left\|\mathbf{e}^{(l)}\right\|_A^2 \,,$$

$$\left\|\mathbf{e}^{(l)}\right\|_A^2 = (\mathbf{A}\mathbf{e}^{(l)})^T\mathbf{e}^{(l)} = \mathbf{r}_l^T\mathbf{A}^{-1}\mathbf{r}_l = \mathbf{B}^{-1}\mathbf{r}^T\mathbf{B}\mathbf{A}^{-1}\mathbf{r}_l \le \lambda_{\max}(\mathbf{B}\mathbf{A}^{-1})\,(\mathbf{B}^{-1}\mathbf{r}_l)^T\mathbf{r}_l \,.$$

available during PCG iteration (4.2.2)

$$\frac{1}{\kappa(\mathbf{B}^{-1}\mathbf{A})}\frac{\left\|\mathbf{e}^{(l)}\right\|_A^2}{\left\|\mathbf{e}^{(0)}\right\|_A^2} \le \frac{(\mathbf{B}^{-1}\mathbf{r}_l)^T\mathbf{r}_l}{(\mathbf{B}^{-1}\mathbf{r}_0)^T\mathbf{r}_0} \le \kappa(\mathbf{B}^{-1}\mathbf{A})\frac{\left\|\mathbf{e}^{(l)}\right\|_A^2}{\left\|\mathbf{e}^{(0)}\right\|_A^2} \tag{4.3.3}$$

$\kappa(\mathbf{B}^{-1}\mathbf{A})$ "'small" ➤ $\mathbf{B}^{-1}$-energy norm of residual $\approx$ $\mathbf{A}$-norm of error !
($\mathbf{r}_l \cdot \mathbf{B}^{-1}\mathbf{r}_l = \mathbf{q}^T\mathbf{r}$ in Algorithm (4.3.2))

△

MATLAB-function: `[x,flag,relr,it,rv] = pcg(A,b,tol,maxit,B,[],x0);`
(A, B may be handles to functions providing $\mathbf{A}\mathbf{x}$ and $\mathbf{B}^{-1}\mathbf{x}$, resp.)

Remark 4.3.8 (Termination criterion in MATLAB-`pcg`).

Implementation (skeleton) of MATLAB built-in `pcg`:

MATLAB PCG algorithm

```
function x = pcg(Afun,b,tol,maxit,Binvfun,x0)
x = x0; r = b - feval(Afun,x); rho = 1;
for i = 1 :  maxit
  y = feval(Binvfun,r);
  rho1 = rho; rho = r' * y;
  if (i == 1)
    p = y;
  else
    beta = rho / rho1;
    p = y + beta * p;
  end
  q = feval(Afun,p);
  alpha = rho /(p' * q);
  x = x + alpha * p;
  if (norm(b - evalf(Afun,x)) <= tolb*norm(b)), return; end
  r = r - alpha * q;
end
```

Dubious termination criterion !

△

Summary:

Advantages of Krylov methods vs. direct elimination (, **IF** they converge at all/sufficiently fast).

- They require system matrix $\mathbf{A}$ in procedural form `y=evalA(x)` $\leftrightarrow \mathbf{y} = \mathbf{A}\mathbf{x}$ only.
- They can perfectly exploit sparsity of system matrix.
- They can cash in on low accuracy requirements (, **IF** viable termination criterion available).
- They can benefit from a good initial guess.

4.4 Essential Skills Learned in Chapter 4

You should know:

- the relation between a linear system of equations with a s.p.d. matrix and a quadratic minimization problem
- how the steepest descendent method works and its convergence properties
- the idea behind the conjugate gradient method and its convergence properties
- how to use the Matlab-function *pcg*
- the importance of the preconditioner