Large-Scale Problems

Small-scale problems:

- "anything goes,"
- no problem to use SVD or other factorizations/decompositions.

Large-scale problems:

- factorizations are not possible in general,
- if possible, use matrix structure (Toeplitz, Kronecker, . . .),
- storage and computing time set the limitations,
- solving, say, the Tikhonov problem for a range of reg. parameters can be a formidable task.

But Wait – There's More

Let us consider the optimization framework for the least-squares problem:

$$\min_x F(x) \ , \qquad F(x) = {}^1\!/_2 \|A\,x - b\|_2^2 \ , \qquad \nabla F(x) = A^T(A\,x - b) \ .$$

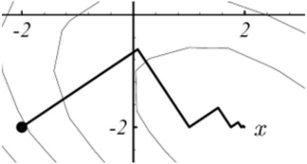
Steepest descent algorithm:

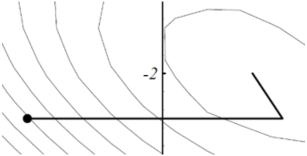
$$x^{[k+1]} = x^{[k]} - \omega_k \, \nabla F(x^{[k]}) = x^{[k]} + \omega_k \, A^T (b - A\,x^{[k]}) \ .$$

CGLS – conjugate gradient algorithm applied to $A^T A\,x = A^T b$:

$$x^{[k+1]} = x^{[k]} - \alpha_k \, d^{[k]} \ , d^{[k]} = \text{search direction}$$

$$\left(d^{[k]}\right)^T A^T A \, d^{[j]} = 0 \ , \qquad j = 1, 2, \ldots, k-1 \ .$$





Advantages of Iterative Methods

We typically think of iterative methods as necessary for solving nonlinear problems. But we can also use them for large, linear problems.

Iterative methods produce a sequence $x^{[0]} \to x^{[1]} \to x^{[2]} \to \cdots$ of iterates that (hopefully) converge to the desired solution, solely through the use of matrix-vector multiplications.

- The matrix A is never altered, only "touched" via matrix-vector multiplications $A\,x$ and $A^T y$.
- The matrix A is not explicitly required – we only need a "black box" that computes the action of A or the underlying operator.
- Atomic operations of iterative methods (mat-vec product, saxpy, norm) suited for high-performance computing.
- Often produce a natural sequence of regularized solutions; stop when the solution is "satisfactory" (parameter choice).

Two Types of Iterative Methods

1. Iterative solution of a regularized problem, such as Tikhonov

$$\left(A^T A + \lambda^2 I\right) x = A^T b \quad \Leftrightarrow \quad \min_x \frac{1}{2} \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2^2 .$$

   Challenges: solve for many λ and needs a good preconditioner!

2. Iterate on the un-regularized system, e.g., on

$$A x = b \quad \text{or} \quad A^T A x = A^T b$$

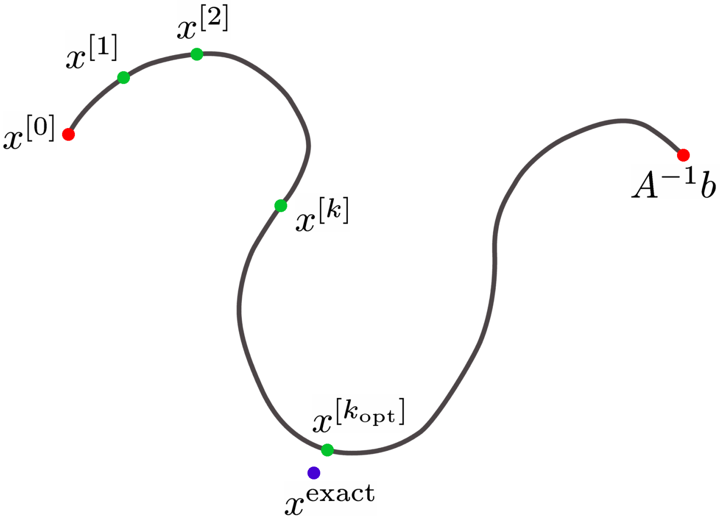
   and use the iteration number as the regularization parameter.

The latter approach relies on *semi-convergence:*

- initial convergence towards the desired x^{exact},
- followed by (slow) convergence to unwanted $A^{-1}b$.

Must stop at the end of the first stage!

Illustration of Semi-Convergence



Landweber Iteration

A classical stationary iterative method:

$$x^{[k]} = x^{[k-1]} + \omega\, A^T (b - A\, x^{[k-1]}) \ , \qquad k = 0, 1, 2, \ldots$$

where $0 < \omega < 2\, \|A^T A\|_2^{-1} = 2\, \sigma_1^{-2}$.

Where does this come from? Consider the function

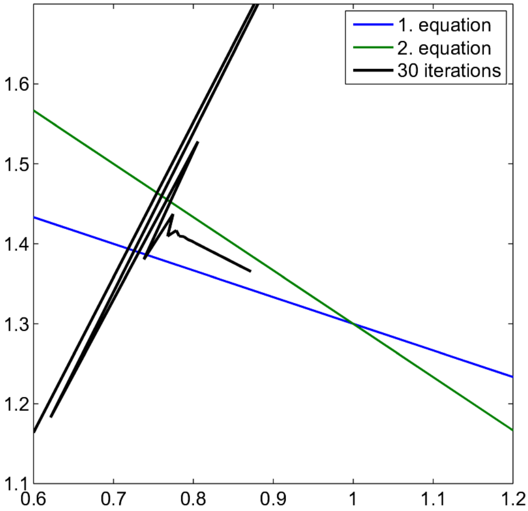
$$\phi(x) = \tfrac{1}{2} \| b - A\, x \|_2^2$$

associated with the least squares problem $\min_x \phi(x)$. It is straightforward (but perhaps a bit tedious) to show that the gradient of ϕ is

$$\nabla \phi(x) = -A^T (b - A\, x).$$

Thus, each step in Landweber's method is a step in the direction of steepest descent. See next slide for an example of iterations.

The Geometry of Landweber Iterations



Towards Convergence Analysis

With an arbitrary starting vector $x^{[0]}$, the kth Landweber iterate is:

$$
\begin{aligned}
x^{[k]} &= x^{[k-1]} + \omega\, A^T \big(b - A\, x^{(k-1)}\big) \\
&= (I - \omega\, A^T A)\, x^{[k-1]} + \omega\, A^T b \\
&= (I - \omega\, A^T A)\, \Big[(I - \omega\, A^T A)\, x^{[k-2]} + \omega\, A^T b\Big] + \omega\, A^T b \\
&= (I - \omega\, A^T A)^2\, x^{[k-2]} + \big((I - \omega\, A^T A) + I\big)\, \omega\, A^T b \\
&= (I - \omega\, A^T A)^3\, x^{[k-3]} + \big((I - \omega\, A^T A)^2 + (I - \omega\, A^T A) + I\big) \omega\, A^T b \\
&= \cdots \\
&= (I - \omega\, A^T A)^k x^{[0]} + \\
&\qquad\qquad \Big[(I - \omega\, A^T A)^{k-1} + (I - \omega\, A^T A)^{k-2} + \cdots + I\Big]\, \omega\, A^T b \\
&= (I - \omega\, A^T A)^k x^{(0)} + \sum_{j=0}^{k-1} (I - \omega\, A^T A)^j\, \omega\, A^T b.
\end{aligned}
$$

SVD Analysis

For simplicity we now assume that $x^{[0]} = 0$. We insert the SVD of the matrix $A = U \Sigma V^T$ and use $I = V V^T$:

$$x^{[k]} = V \sum_{j=0}^{k-1} (I - \omega \Sigma^2)^j \omega \Sigma U^T b = V \Phi^{(k)} \Sigma^{-1} U^T b,$$

where we introduced the $n \times n$ diagonal matrix

$$\Phi^{(k)} = \sum_{j=0}^{k-1} (I - \omega \Sigma^2)^j \omega \Sigma^2 = \omega \Sigma^2 \sum_{j=0}^{k-1} (I - \omega \Sigma^2)^j = \begin{pmatrix} \phi_1^{(k)} & & \\ & \phi_2^{(k)} & \\ & & \ddots \end{pmatrix}$$

with diagonal elements

$$\phi_i^{(k)} = \omega \sigma_i^2 \sum_{j=0}^{k-1} (1 - \omega \sigma_i^2)^j, \qquad i = 1, 2, \ldots, n.$$

The Filter Factors

The sum $\sum_{j=0}^{k-1}(1-\omega\,\sigma_i^2)^j$ is a geometric series,

$$\sum_{j=0}^{k-1} z^j = (1-z^k)/(1-z) \ ,$$

and thus for $i = 1, 2, \ldots, n$ we have

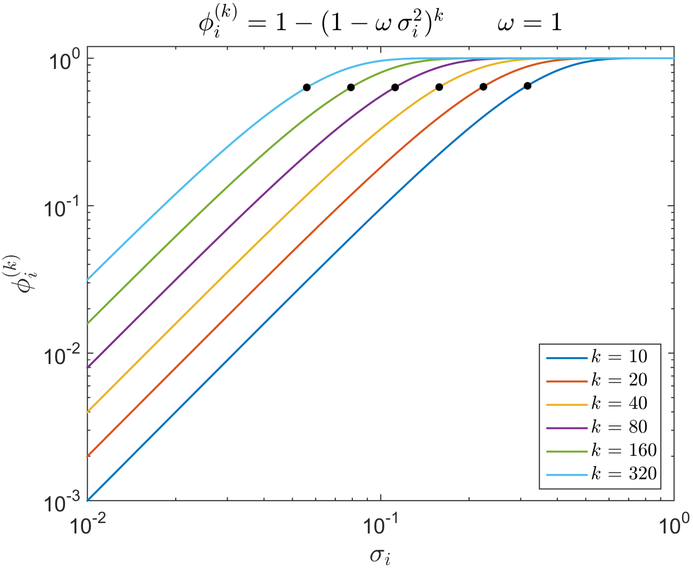
$$\phi_i^{(k)} = \omega\,\sigma_i^2 \sum_{j=0}^{k-1}(1-\omega\,\sigma_i^2)^j = \omega\,\sigma_i^2\,\frac{1-(1-\omega\,\sigma_i^2)^k}{1-(1-\omega\,\sigma_i^2)} = 1-(1-\omega\,\sigma_i^2)^k.$$

Let $\sigma_{\text{break}}^{(k)}$ denote the value of σ_i for which $\phi_i^{(k)} = 0.5$. Then

$$\frac{\sigma_{\text{break}}^{(k)}}{\sigma_{\text{break}}^{(2k)}} = \sqrt{1+\left(\tfrac{1}{2}\right)^{\frac{1}{2k}}} \to \sqrt{2} \quad \text{for} \quad k \to \infty.$$

Hence, as k increases, the breakpoint tends to be reduced by a factor $\sqrt{2} \approx 1.4$ each time the number of iterations k is doubled.

Landweber Filter Factors



Cimmino Iteration

Cimmino's method is a variant of Landweber's method, with a diagonal scaling:

$$x^{[k]} = x^{[k-1]} + \omega\, A^T D\, (b - A\, x^{[k-1]}), \qquad k = 1, 2, \ldots$$

in which $D = \mathrm{diag}(d_i)$ is a diagonal matrix whose elements are defined in terms of the rows $a_i^T = A(i, :)$ of A as

$$d_i = \begin{cases} \dfrac{1}{m} \dfrac{1}{\|a_i\|_2^2}, & a_i \neq 0 \\ 0, & a_i = 0. \end{cases}$$

Cimmino's method may often converge faster than Landweber.

. . . and the prize for best acronym goes to "ART"

Kaczmarz's method = algebraic reconstruction technique (ART).

Let $a_i^T = A(i,:) =$ ith row of A, and $b_i =$ ith component b.

Each iteration of ART involves the following "sweep" over all rows:

$$z^{(0)} = x^{[k-1]}$$

for $i = 1, \ldots, m$

$$z^{(i)} = z^{(i-1)} + \frac{b_i - a_i^T z^{(i-1)}}{\|a_i\|_2^2}\, a_i$$

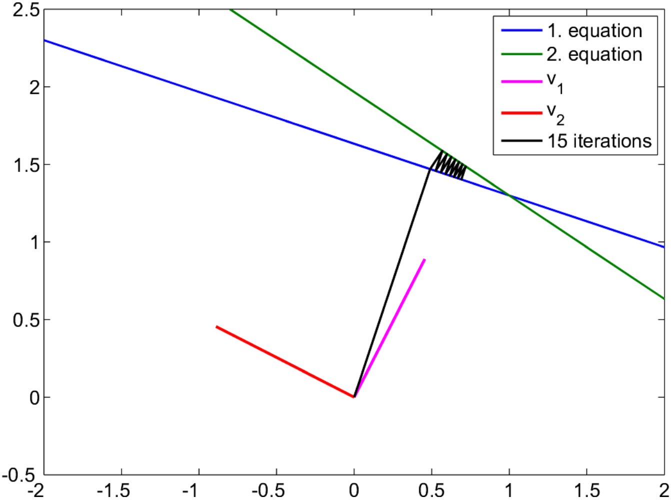
end

$$x^{[k]} = z^{(m)}$$

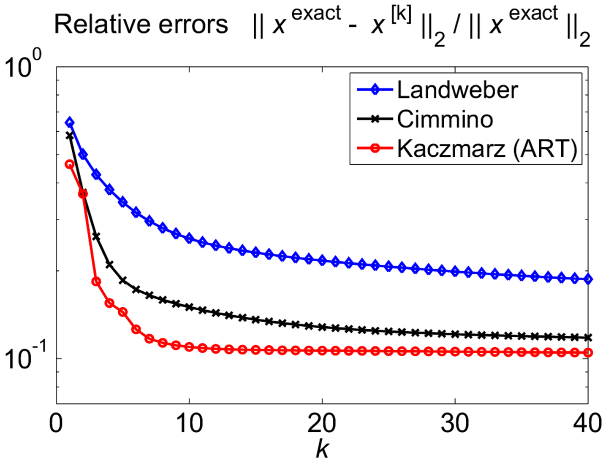
This method is not "simultaneous" because each row must be processed sequentially.

In general: fast initial convergence, then slow. See next slides.

The Geometry of ART Iterations



Slow Convergence of SIRT and ART Methods



The test problem is `shaw`.

Projection Methods

As an important step towards the faster *Krylov subspace methods*, we consider projection methods.

Assume the columns of $W_k = (w_1, \ldots, w_k) \in \mathbb{R}^{n \times k}$ form a "good basis" for an approximate regularized solution, obtained by solving

$$\min_x \|A\,x - b\|_2 \qquad \text{s.t.} \qquad x \in \mathcal{W}_k = \text{span}\{w_1, \ldots, w_k\}.$$

This solution takes the form

$$x^{(k)} = W_k\, y^{(k)}, \qquad y^{(k)} = \text{argmin}_y \|(A\,W_k)\,y - b\|_2,$$

and we refer to the least squares problem $\|(A\,W_k)\,y - b\|_2$ as the *projected problem*, because it is obtained by projecting the original problem onto the k-dimensional subspace $\text{span}(w_1, \ldots, w_k)$.

If $W_k = V_k$ then we obtain the TSVD method, and $x^{(k)} = x_k$

But we want to work with computationally simpler basis vectors.

Computations with DCT Basis

Note that

$$\widehat{A}_k = A\,W_k = (W_k^T A^T)^T = \Big[(W^T A^T)^T\Big]_{:,1:k} .$$

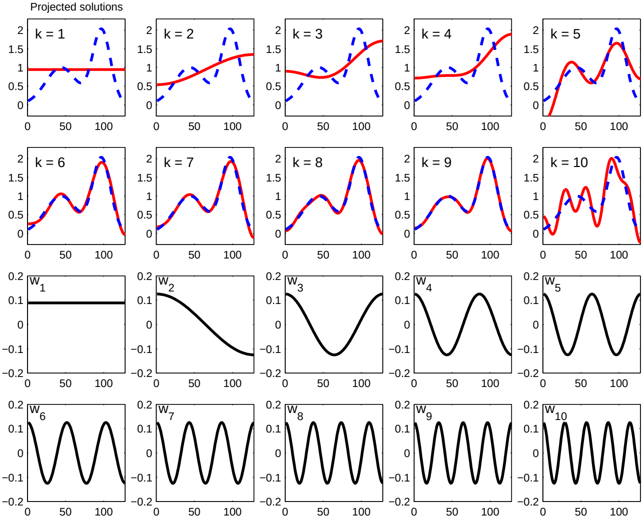
In the case of the discrete cosine basis, multiplication with W^T is equivalent to a DCT. The algorithm takes the form:

```
Akhat = dct(A')';
Akhat = Akhat(:,1:k);
y = Akhat\b;
xk = idct([y;zeros(n-k,1)]);
```

Next page:

- Top: solutions $x^{(k)}$ for $k = 1, \ldots, 10$.
- Bottom: cosine basis w_i, $i = 1, \ldots, 10$.

Example Using Discrete Cosine Basis (`shaw`)



The Krylov Subspace

The **Krylov subspace**, defined as

$$\mathcal{K}_k \equiv \text{span}\{A^T b, A^T A\, A^T b, (A^T A)^2 A^T b, \ldots, (A^T A)^{k-1} A^T b\},$$

always *adapts* itself to the problem at hand! But the "naive" basis $q_i = (A^T A)^{i-1} A^T b$ is NOT useful due to scaling issues.

The normalized, "naive" basis

$$p_i = (A^T A)^{i-1} A^T b \,/\, \|(A^T A)^{i-1} A^T b\|_2, \qquad i = 1, 2, \ldots$$

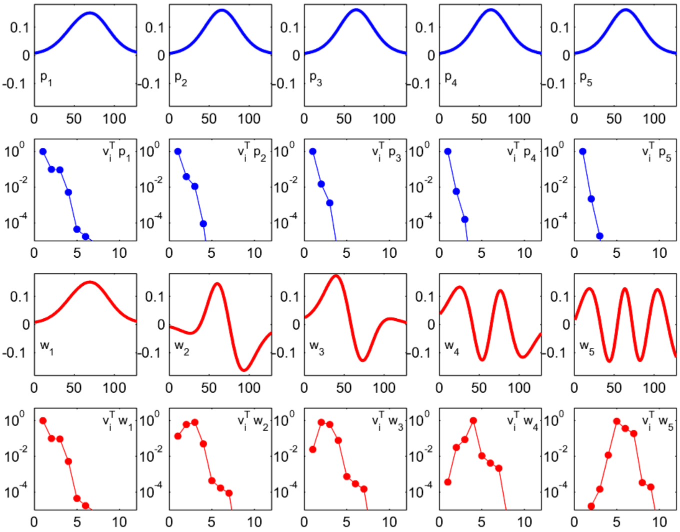
is NOT useful either: $p_i \to v_1$ as $i \to \infty$. See the next slide.

Moreover, the condition numbers of the matrices $[q_1, \ldots, q_k]$ and $[p_1, \ldots, p_k]$ increases dramatically with k

Use modified Gram-Schmidt for which $\text{cond}([w_1, \ldots, w_k]) = 1$:

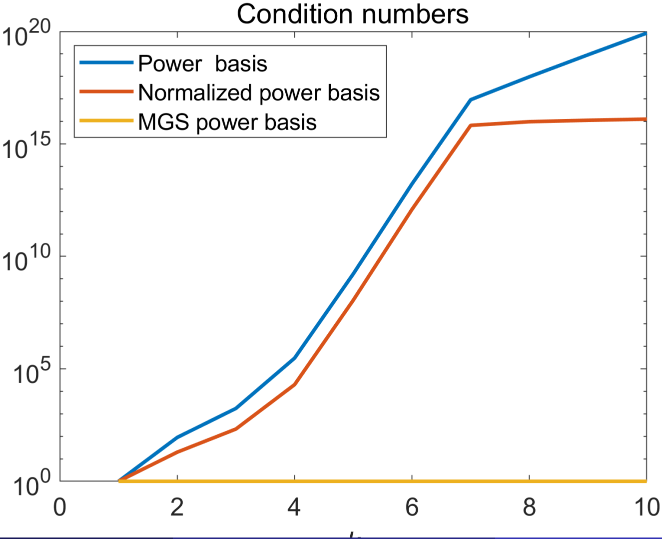
$$\begin{array}{lll}
w_1 \leftarrow A^T b; & w_1 \leftarrow w_1 / \|w_1\|_2 & \\
w_2 \leftarrow A^T A\, w_1; & w_2 \leftarrow w_2 - w_1^T w_2\, w_1; & w_2 \leftarrow w_2 / \|w_2\|_2 \\
w_3 \leftarrow A^T A\, w_2; & w_3 \leftarrow w_3 - w_1^T w_3\, w_1; & \\
 & w_3 \leftarrow w_3 - w_2^T w_3\, w_2; & w_3 \leftarrow w_3 / \|w_3\|_2
\end{array}$$

Comparison of basis vectors p_i (blue) and w_i (red)



Conditioning of the bases

This figure shows the condition numbers of the three matrices of basis vectors $[q_1, \ldots, q_k]$ and $[p_1, \ldots, p_k]$ and $[w_1, \ldots, w_k]$ for increasing k.



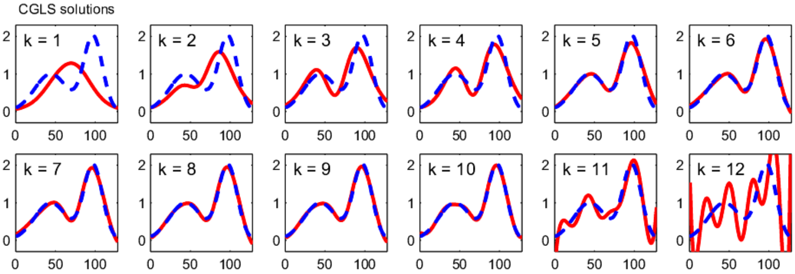
Can We Compute $x^{(k)}$ Without Storing W_k?

Yes: the CGLS algorithm – see next slide – computes iterates given by

$$x^{(k)} = \text{argmin}_x \, \|A\,x - b\|_2 \qquad \text{s.t.} \qquad x \in \mathcal{K}_k .$$

The algorithm eventually converges to the least squares solution.

But since $\mathcal{K}_k$ is a good subspace for approximate regularized solutions, CGLS exhibits semi-convergence.



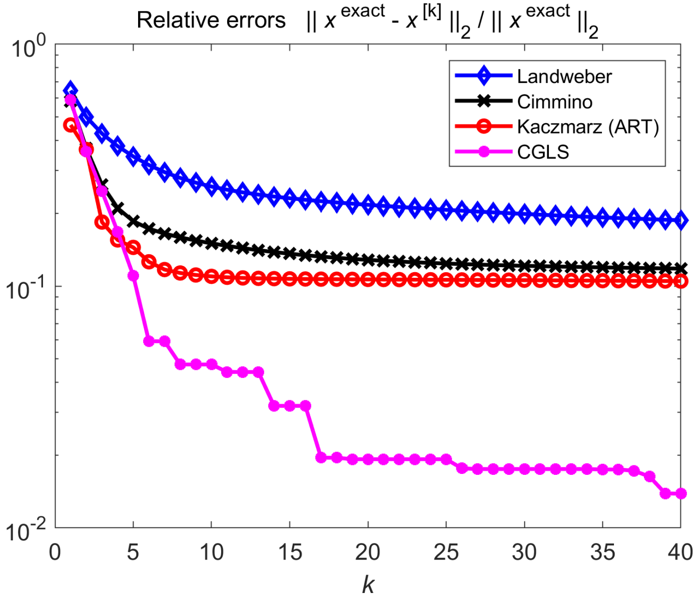
CGLS = Conjugate Gradients for Least Squares

The CGLS algorithm for solving $\min_x \|A\,x - b\|_2$ takes the following form:

$$
\begin{aligned}
& x^{(0)} = \text{starting vector (e.g., zero)} \\
& r^{(0)} = b - A\,x^{(0)} \\
& d^{(0)} = A^T r^{(0)} \\
& \text{for } k = 1, 2, \ldots \\
& \qquad \bar{\alpha}_k = \|A^T r^{(k-1)}\|_2^2 / \|A\,d^{(k-1)}\|_2^2 \\
& \qquad x^{(k)} = x^{(k-1)} + \bar{\alpha}_k\, d^{(k-1)} \\
& \qquad r^{(k)} = r^{(k-1)} - \bar{\alpha}_k\, A\, d^{(k-1)} \\
& \qquad \bar{\beta}_k = \|A^T r^{(k)}\|_2^2 / \|A^T r^{(k-1)}\|_2^2 \\
& \qquad d^{(k)} = A^T r^{(k)} + \bar{\beta}_k\, d^{(k-1)} \\
& \text{end}
\end{aligned}
$$

For Tikhonov, just replace A and b with $\begin{pmatrix} A \\ \lambda I \end{pmatrix}$ and $\begin{pmatrix} b \\ 0 \end{pmatrix}$.

Comparison of CGLS With the Previous Methods



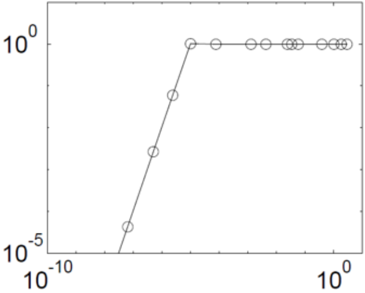
SVD Analysis – Outside the Scope of This Course

It is pretty hairy, but we can perform an SVD analysis along these lines:

$$\phi_i^{(k)} = 1 - \prod_{j=1}^{k} \frac{\theta_j^{(k)} - \sigma_i^2}{\theta_j^{(k)}} = \text{filter factors}$$

$$\theta_k^{(k)} = \text{eigenvalalues of } A^T A \text{ projected on } \mathcal{K}_k$$

$$\mathcal{K}_k = \text{span}\{A^T b, (A^T A) A^T b, \ldots, (A^T A)^{k-1} A^T b\} = \text{Krylov subspace}$$



Other Iterations – GMRES and RRGMRES

Sometimes difficult or inconvenient to write a matrix-free black-box function for multiplication with A^T. Can we avoid this?

The GMRES method for square nonsymmetric matrices is based on the Krylov subspace

$$\mathcal{K}_k = \mathrm{span}\{b, Ab, A^2 b, \ldots, A^{k-1} b\}.$$

The presence of the noisy data $b = b^{\mathrm{exact}} + e$ in this subspace is unfortunate: the solutions include the noise component e!

A better subspace, underlying the RRGMRES method:

$$\vec{\mathcal{K}}_k = \mathrm{span}\{A\, b, A^2\, b, \ldots, A^k\, b\}.$$

Now the noise vector is multiplied with A (smoothing) at least once.

Symmetric matrices: use MR-II (a simplified variant).