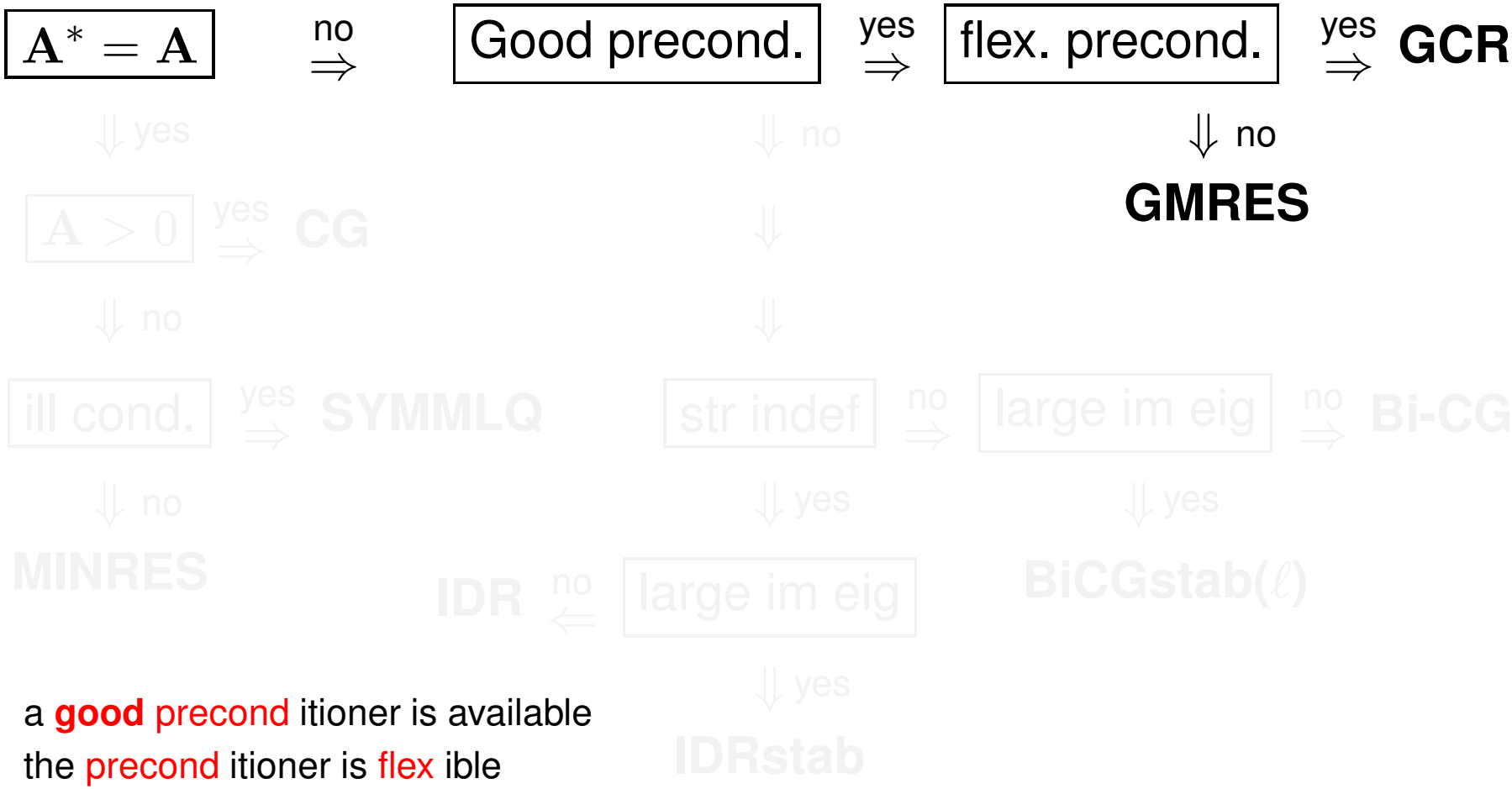
# Numerical Linear Algebra

## Krylov methods for Hermitian systems

Gerard Sleijpen and Martin van Gijzen

November 8, 2017

# Solving $\mathrm{Ax} = \mathrm{b}$, an overview




a **good** precond itioner is available

the precond itioner is flex ible

$\mathbf{A} + \mathbf{A}^*$ is strongly indefinite

$\mathbf{A}$ has large imaginary eigenvalues

# Solving $\mathrm{Ax} = \mathrm{b}$, an overview

$\boxed{\mathbf{A}^* = \mathbf{A}}$ $\overset{\text{no}}{\Rightarrow}$ Good precond. $\overset{\text{yes}}{\Rightarrow}$ flex. precond. $\overset{\text{yes}}{\Rightarrow}$ **GCR**

$\Downarrow$ yes

$\boxed{\mathbf{A} > 0}$ $\overset{\text{yes}}{\Rightarrow}$ **CG**

$\Downarrow$ no

ill cond. $\overset{\text{yes}}{\Rightarrow}$ **SYMMLQ**

$\Downarrow$ no

**MINRES**

$\Downarrow$ no

**GMRES**

a **good** precond itioner is available

the precond itioner is flex ible

$\mathbf{A} + \mathbf{A}^*$ is strongly indefinite

$\mathbf{A}$ has large imaginary eigenvalues

# Program Lecture 7

- Computing a basis for the Krylov subspace
  - The Arnoldi basis
  - Lanczos method
- Solution methods for linear systems
  - Conjugate gradients
  - Minres, Conjugate Residuals
  - SYMMLQ

# Projection methods

Last time you saw a general framework to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ with a projection method. The main steps are:

- Use recursion to compute a basis $\mathbf{V}_k$ for the search subspace;
- Project the problem and compute it's representation w.r.t. the basis (gives low dimensional problem);
- Solve projected problem (gives low dimensional solution vector $\vec{y}_k$);
- Compute (high dimensional) solution $\mathbf{x}_k = \mathbf{V}_k \, \vec{y}_k$.

Today we will see how symmetry (Hermitian problems) can be exploited to enhance efficiency.

# The Krylov subspace

The subspace $\text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\}$ is called the **Krylov subspace** of **order** $k$, generated by the matrix $\mathbf{A}$ and initial vector $\mathbf{r}_0$ and is denoted by

$$\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\}$$

Projection methods that search Krylov subspaces are called **Krylov subspace methods**.

As we have seen in the previous lessons, a stable orthogonal basis for $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ can be computed using Arnoldi's method.

# Arnoldi's method

Choose a starting vector $\mathbf{v}_1$ with $\|\mathbf{v}_1\|_2 = 1$.

For $k = 1, \ldots,$ do % *iteration*

  $\mathbf{w} = \mathbf{A}\mathbf{v}_k$ % *expansion*

  For $i = 1, \ldots, k,$ do % *orthogonalisation*

    $h_{i,k} = \mathbf{v}_i^* \mathbf{w}$

    $\mathbf{w} = \mathbf{w} - h_{i,k}\mathbf{v}_i$

  end for

  $h_{k+1,k} = \|\mathbf{w}\|_2$

  if $h_{k+1,k} = 0,$ stop % *invariant subspace spanned*

  $\mathbf{v}_{k+1} = \mathbf{w}/h_{k+1,k}$ % *new basis vector*

end for

# The Arnoldi relation

The Arnoldi method can be summarised in a compact way. With

$$\underline{H}_k \equiv \begin{bmatrix} h_{1,1} & \cdots & \cdots & h_{1,k} \\ h_{2,1} & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ & & h_{k,k-1} & h_{k,k} \\ O & & & h_{k+1,k} \end{bmatrix}$$

and $\mathbf{V}_k \equiv [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$, we have the **Arnoldi relation**

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_{k+1}\underline{H}_k$$

# A is Hermitian

Let $\mathbf{A}$ be Hermitian.

According to the Arnoldi relation we have

$$\mathbf{V}_k^* \mathbf{A} \mathbf{V}_k = \mathbf{V}_k^* \mathbf{V}_{k+1} \underline{H}_k = H_k,$$

where $H_k$ is the $k \times k$ upper block of $\underline{H}_k$.

Moreover, if $\mathbf{A}$ is Hermitian we have

$$H_k^* = \mathbf{V}_k^* \mathbf{A}^* \mathbf{V}_k = \mathbf{V}_k^* \mathbf{A} \mathbf{V}_k = H_k.$$

So $H_k$ is Hermitian and upper Hessenberg.

This implies that $H_k$ must be tridiagonal.

# A is Hermitian (2)

So

$$H_k = \begin{bmatrix} h_{1,1} & h_{1,2} & & O \\ h_{2,1} & \ddots & \ddots & \\ & \ddots & \ddots & h_{k-1,k} \\ O & & h_{k,k-1} & h_{k,k} \end{bmatrix}.$$

With $\alpha_k \equiv h_{k,k} = \overline{h_{k,k}}$ and $\beta_{k+1} \equiv h_{k+1,k} = \overline{h_{k,k+1}}$ the Arnoldi method simplifies to the (Hermitian) **Lanczos method**.

# A is Hermitian (2)

So

$$H_k = \begin{bmatrix} \alpha_1 & \bar{\beta}_2 & & O \\ \beta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_k \\ O & & \overline{\beta_k} & \alpha_{k,k} \end{bmatrix}.$$

With $\alpha_k \equiv h_{k,k} = \overline{h_{k,k}}$ and $\beta_{k+1} \equiv h_{k+1,k} = \overline{h_{k,k+1}}$ the Arnoldi method simplifies to the (Hermitian) **Lanczos method**. With the Lanczos method it is possible to compute a new orthonormal basis vector using only the two previous basis vectors.

# Lanczos' method

Choose a starting vector $\mathbf{v}_1$ with $\|\mathbf{v}_1\|_2 = 1$

$\beta_1 = 0,\ \mathbf{v}_0 = 0$ % *Initialization*

For $k = 1, \ldots$ do % *Iteration*

$\alpha_k = \mathbf{v}_k^* \mathbf{A} \mathbf{v}_k$

$\mathbf{w} = \mathbf{A}\mathbf{v}_k - \alpha_k \mathbf{v}_k - \beta_k \mathbf{v}_{k-1}$

% *New direction orthogonal to the previous* $\mathbf{v}$*s*

$\beta_{k+1} = \|\mathbf{w}\|_2$

$\mathbf{v}_{k+1} = \mathbf{w}/\beta_{k+1}$ % *Normalization*

end for

Note that $\beta_k > 0$.

# Lanczos' method

Let

$$\underline{T}_k \equiv \begin{bmatrix} \alpha_1 & \beta_2 & & & 0 \\ \beta_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_k \\ 0 & & & \beta_k & \alpha_k \\ & & & 0 & \beta_{k+1} \end{bmatrix}.$$

and $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$. Then, the **Lanczos relation** reads

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_{k+1}\underline{T}_k.$$

# Eigenvalue methods

Arnoldi's and Lanczos' method were originally proposed as iterative methods to compute the eigenvalues of a matrix $\mathbf{A}$:

$$\mathbf{V}_k^* \mathbf{A} \mathbf{V}_k = H_k$$

is 'almost' a similarity transformation.

The eigenvalues of $H_k$ are called **Ritz values** (of $\mathbf{A}$ of order $k$).

$$H_k \vec{z} = \theta \vec{z} \quad \Leftrightarrow \quad \mathbf{A}\mathbf{u} - \theta \mathbf{u} \perp \mathbf{V}_k \quad \text{where } \mathbf{u} \equiv \mathbf{V}_k \vec{z}.$$

$\mathbf{u}$ is a **Ritz vector**, $(\theta, \mathbf{u})$ is a **Ritz pair**.

# Optimal approximations

The Lanczos method provides a cheap way to compute an orthogonal basis for the Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$.
Our approximations can be written as

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \vec{y}_k,$$

where $\vec{y}_k$ is determined so that either the error

$$\|\mathbf{x} - \mathbf{x}_k\|_A = \sqrt{(\mathbf{x} - \mathbf{x}_k)^* \mathbf{A} (\mathbf{x} - \mathbf{x}_k)}$$

is minimised in $\mathbf{A}$-norm (only meaningful if $\mathbf{A}$ is pos. def.) or that

$$\|\mathbf{r}_k\|_2 = \|\mathbf{A}(\mathbf{x} - \mathbf{x}_k)\|_2 = \sqrt{\mathbf{r}_k^* \mathbf{r}_k},$$

is minimised, i.e. the norm of the residual is minimised.

# Optimal approximations (2)

We first look at the minimisation of the error in the $A$-norm:

$$\|\mathbf{x} - \mathbf{x}_0 - \mathbf{V}_k \vec{y}_k\|_A$$

is minimal iff $\mathbf{e}_k \equiv \mathbf{x} - \mathbf{x}_0 - \mathbf{V}_k \vec{y}_k \perp_A \mathbf{V}_k$, or, equivalently, $\mathbf{V}_k \perp \mathbf{A}\mathbf{e}_k = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_k \vec{y}_k$. This yields

$$\mathbf{V}_k^* \mathbf{A} \mathbf{V}_k \vec{y}_k = \mathbf{V}_k^* \mathbf{r}_0.$$

With $T_k \equiv \mathbf{V}_k^* \mathbf{A} \mathbf{V}_k$, the $k \times k$ upper block of $\underline{T}_k$, and $\mathbf{r}_0 = \|\mathbf{r}_0\|_2 \, \mathbf{v}_1$, we get

$$T_k \vec{y}_k = \|\mathbf{r}_0\|_2 \, e_1$$

with $e_1$ the first canonical basis vector.

# Optimal approximations (3)

In particular, we have that the residuals are orthogonal to the basis vectors:

$$\mathbf{r}_k = \mathbf{A}\mathbf{e}_k = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_k \vec{y}_k \perp \mathbf{V}_k$$

Since the $\mathbf{r}_k$'s are orthogonal, each residual is just a multiple of the corresponding basis vector $\mathbf{v}_{k+1}$: $\mathbf{r}_k$ and $\mathbf{v}_{k+1}$ are **collinear**. (Recall that $\mathbf{r}_0 = \|\mathbf{r}_0\|_2 \, \mathbf{v}_1$.)

This also means that the residuals form an orthogonal Krylov basis for the Krylov subspace.

# Towards a practical algorithm

The main problem in the Lanczos algorithm is that all $\mathbf{v}_i$ (or $\mathbf{r}_{i-1}$) have to be stored to compute $\mathbf{x}_k$. This problem can be overcome by making an implicit $LU$-factorisation, $T_k = L_k U_k$ of $T_k$, and updating $\mathbf{x}_k$,

$$\mathbf{x}_k = \mathbf{x}_0 + \left(\mathbf{V}_k U_k^{-1}\right)\left(L_k^{-1}(\|\mathbf{r}_0\|_2\, e_1)\right),$$

in every iteration.

Details can be found in the book of Van der Vorst.

With this technique we get the famous and very elegant **Conjugate Gradient** method.

# The Conjugate Gradient method

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0,\ \mathbf{u}_{-1} = \mathbf{0},\ \rho_{-1} = 1$ % *Initialization*

For $k = 0, 1, \ldots,$ do

$\rho_k = \mathbf{r}_k^* \mathbf{r}_k,\ \beta_k = \rho_k / \rho_{k-1}$

$\mathbf{u}_k = \mathbf{r}_k + \beta_k \mathbf{u}_{k-1}$ % *Update direction vector*

$\mathbf{c}_k = \mathbf{A}\mathbf{u}_k$

$\sigma_k = \mathbf{u}_k^* \mathbf{c}_k,\ \alpha_k = \rho_k / \sigma_k$

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{u}_k$ % *Update iterate*

$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{c}_k$ % *Update residual*

end for

Note that, in our notation, the CG $\alpha_j$ and $\beta_j$ are not the same as the Lanczos $\alpha_j$ and $\beta_j$.

# The Conjugate Gradient method

$\mathbf{x} = \mathbf{x}_0,\ \mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x},\ \mathbf{u} = \mathbf{0},\ \rho = 1$ % *Initialization*

For $k = 0, 1, \ldots,$ $k_{\max}$ do

$\rho' = \rho,\ \rho = \mathbf{r}^*\mathbf{r},\ \beta = \rho/\rho'$

Stop if $\rho \leq$ tol

$\mathbf{u} \leftarrow \mathbf{r} + \beta\,\mathbf{u}$ % *Update direction vector*

$\mathbf{c} = \mathbf{A}\mathbf{u}$

$\sigma = \mathbf{u}^*\mathbf{c},\ \alpha = \rho/\sigma$

$\mathbf{x} \leftarrow \mathbf{x} + \alpha\,\mathbf{u}$ % *Update iterate*

$\mathbf{r} \leftarrow \mathbf{r} - \alpha\,\mathbf{c}$ % *Update residual*

end for

# Properties of CG

CG has several favourable properties:

- The method uses limited memory: only three vectors need to be stored;
- The method is optimal: the error is minimised in $\mathbf{A}$-norm;
- The method is finite: the $(n+1)$st residual must be zero since all the residuals are orthogonal;
- The method is robust (if $\mathbf{A}$ is HPD): $\sigma_k \equiv \mathbf{u}_k^* \mathbf{A} \mathbf{u}_k = 0$ and $\rho_k \equiv \mathbf{r}_k^* \mathbf{r}_k = 0$ both imply that the true solution has been found (that $\mathbf{r}_k = \mathbf{0}$).
- $\mathbf{u}_i^* \mathbf{A} \mathbf{u}_j = 0$ and $\mathbf{r}_i^* \mathbf{r}_j = 0$ for $i \neq j$

# Lanczos matrix and CG

Since CG and Lanczos are mathematically equivalent it should be possible to recover the Lanczos matrix $T_k$ from the CG-iteration parameters. This is indeed the case

$$T_k = \begin{bmatrix} \frac{1}{\alpha_0} & \frac{\sqrt{\beta_1}}{\alpha_0} & & & 0 \\ \frac{\sqrt{\beta_1}}{\alpha_0} & \frac{1}{\alpha_1} + \frac{\beta_1}{\alpha_0} & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \frac{\sqrt{\beta_{k-1}}}{\alpha_{k-2}} \\ 0 & & & \frac{\sqrt{\beta_{k-1}}}{\alpha_{k-2}} & \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-1}}{\alpha_{k-2}} \end{bmatrix}.$$

Note that the CG $\beta_0$ is not in the matrix ($\beta_0$ is meaningless anyway since $\mathbf{u}_{-1} = \mathbf{0}$).

# Conjugate Gradient Error Analysis

Since $\mathbf{e}_k \equiv \mathbf{x} - \mathbf{x}_k = \mathbf{e}_0 - \mathbf{V}_k \vec{y}_k$ with $\mathbf{V}_k \vec{y}_k \in \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$, and $\mathbf{r}_0 = \mathbf{A}\mathbf{e}_0$, we see that

$$\mathbf{e}_k = \mathbf{e}_0 - \gamma_1 \mathbf{A}\mathbf{e}_0 - \gamma_2 \mathbf{A}^2 \mathbf{e}_0 - \cdots - \gamma_k \mathbf{A}^k \mathbf{e}_0 = p_k(\mathbf{A})\mathbf{e}_0$$

So, $\mathbf{e}_k$ is a degree $k$ polynomial $p_k$ in $\mathbf{A}$ times $\mathbf{e}_0$, with $p_k(0) = 1$.
CG minimises $\|\mathbf{e}_k\|_A = \|\mathbf{e}_0 - \mathbf{V}_k \vec{y}_k\|_A$ with $\mathbf{V}_k \vec{y}_k \in \mathcal{K}_k(A, r_0)$.
Consequently,

$$\|\mathbf{e}_k\|_A = \|p_k(\mathbf{A})\mathbf{e}_0\|_A \leq \|\tilde{p}_k(\mathbf{A})\mathbf{e}_0\|_A$$

for all degree $k$ polynomials $\tilde{p}_k$ with $\tilde{p}_k(0) = 1$. Here we used that $\tilde{p}_k(\mathbf{A})\mathbf{e}_0$ is also an error, i.e., of the form $\tilde{p}_k(\mathbf{A})\mathbf{e}_0 = \mathbf{e}_0 - \mathbf{V}_k \tilde{y}_k$ .

# A convergence bound for CG

The iterates $\mathbf{x}_k$ obtained from the CG algorithm satisfy the following inequality:

$$\frac{\|\mathbf{x}-\mathbf{x}_k\|_A}{\|\mathbf{x}-\mathbf{x}_0\|_A} \leq 2\left(\frac{\sqrt{\mathcal{C}_2}-1}{\sqrt{\mathcal{C}_2}+1}\right)^k \leq 2\exp\left(-\frac{2k}{\sqrt{\mathcal{C}_2}}\right).$$

$\mathcal{C}_2$ is the 2-condition number of $\mathbf{A}$, which is for HPD-matrices

$$\mathcal{C}_2 \equiv \mathcal{C}_2(\mathbf{A}) = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$$

# Proof (sketch)

The error $\mathbf{e}_k = \mathbf{x} - \mathbf{x}_k$ can be written as $p_k(\mathbf{A})\mathbf{e}_0$ with $p_k$ a polynomial such that $p_k(0) = 1$. Hence, since CG is optimal,

$$\|\mathbf{e}_k\|_A = \|p_k(\mathbf{A})\mathbf{e}_0\|_A \leq \|\tilde{p}_k(\mathbf{A})\mathbf{e}_0\|_A \qquad \forall\ \tilde{p}_k \text{ with } \tilde{p}_k(0) = 1.$$

Since, in this Hermitian case, there is an orthonormal basis of eigenvectors of $\mathbf{A}$, we have

$$\|\tilde{p}_k(\mathbf{A})\mathbf{e}_0\|_A \leq \max_i |\tilde{p}_k(\lambda_i(\mathbf{A}))| \cdot \|\mathbf{e}_0\|_A$$

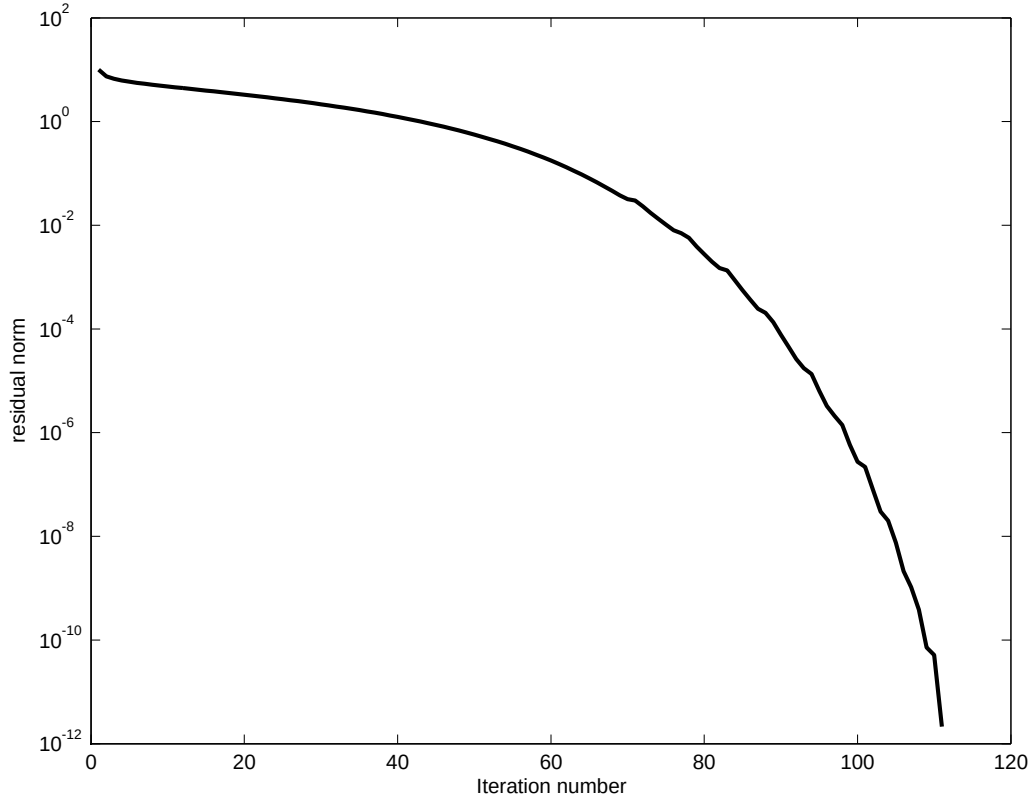
The convergence bound can now be proved by taking for $\tilde{p}_k$ a scaled Chebyshev polynomial that is transformed (shifted and scaled) to the interval $[\lambda_{\text{min}}, \lambda_{\text{max}}]$.

# Superlinear convergence

The upperbound on the CG-error is in practice very pessimistic.

Typically the rate of convergence increases during the process.

This is called **superlinear convergence**.



# Superlinear convergence (2)

Zeros of the CG polynomial $p_k$ are Ritz values
(of $\mathbf{A}$ of order $k$, i.e., the eigenvalue of $T_k$).

*Proof.* If $p_k(\theta) = 0,$ then

$$p_k(\lambda) = (\theta - \lambda)\, q(\lambda) \quad \text{for some } k-1 \text{ degree polynimial } q.$$

Hence,

$$\mathbf{e}_k = (\theta\,\mathbf{I} - \mathbf{A})\, q(\mathbf{A})\mathbf{e}_0 \quad \Rightarrow \quad \mathbf{r}_k = (\theta\,\mathbf{I} - \mathbf{A})\, q(\mathbf{A})\mathbf{r}_0. \qquad (*)$$

Since $\mathbf{u} \equiv q(\mathbf{A})\mathbf{r}_0 = \mathbf{V}_k\, \vec{z}_k$ for some $k$-vector $\vec{z}_k$, we have that

$$(\theta\,\mathbf{I} - \mathbf{A})\,\mathbf{u} = \mathbf{A}\mathbf{e}_k = \mathbf{r}_k \perp \mathbf{V}_k \quad \text{and} \quad \mathbf{u} \in \text{span}(\mathbf{V}_k) = \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0),$$

whence $(\theta, \mathbf{u})$ is a Ritz pair.

A counting argument now shows that Ritz values are zeros of the CG polynomial.

# Superlinear convergence (3)

Superlinear convergence occurs when Ritz values converge to extreme eigenvalues. Then the component in the direction of that eigenvector is found.

> *Explanation. If, in addition,* $\theta \approx \lambda_j$, *then* $(*)$ *shows that,*
> *the eigenvector* $\mathbf{w}_j$ *of* $\mathbf{A}$ *associated with* $\lambda_j$, *i.e.,* $\mathbf{A}\mathbf{w}_j = \lambda_j \mathbf{w}_j$,
> *is* $(\approx)$ *'deflated' from* $\mathbf{e}_k$ *and from* $\mathbf{r}_k = \mathbf{A}\mathbf{e}_k$.

That is, both the error and the residual have component $(\approx)$ zero in the direction of that eigenvector $(\mathbf{w}_j)$.

Note. Assume, $0 < \lambda_1 < \lambda_2 < \ldots < \lambda_n$. When is $\theta$ sufficiently close to $\lambda_1$? It can be shown that superlinear convergence is noticeable already if $\theta < \lambda_2$.

# Superlinear convergence (3)

Superlinear convergence occurs when Ritz values converge to extreme eigenvalues. Then the component in the direction of that eigenvector is found.

*Explanation. If, in addition,* $\theta \approx \lambda_j$, *then* $(*)$ *shows that,*
*the eigenvector* $\mathbf{w}_j$ *of* $\mathbf{A}$ *associated with* $\lambda_j$, *i.e.,* $\mathbf{A}\mathbf{w}_j = \lambda_j \mathbf{w}_j$,
*is* $(\approx)$ *'deflated' from* $\mathbf{e}_k$ *and from* $\mathbf{r}_k = \mathbf{A}\mathbf{e}_k$.

Convergence of CG from then on is determined by the reduced spectrum from which converged eigenvalues have been removed.

*Explanation.* $\|p_{k+\ell}(\mathbf{A})\mathbf{e}_0\|_A \leq \|\tilde{p}_\ell(\mathbf{A})\mathbf{e}_k\|_A = \|\tilde{p}_\ell(\widetilde{\mathbf{A}})\mathbf{e}_k\|_{\tilde{A}}$.

*With the* ***deflated matrix*** $\widetilde{\mathbf{A}} \equiv (1 - \frac{\mathbf{w}_j \mathbf{w}_j^*}{\mathbf{w}_j^* \mathbf{w}_j})\mathbf{A}$, *we used that* $\mathbf{A}\mathbf{e}_k = \widetilde{\mathbf{A}}\mathbf{e}_k$.

# Superlinear convergence (4)

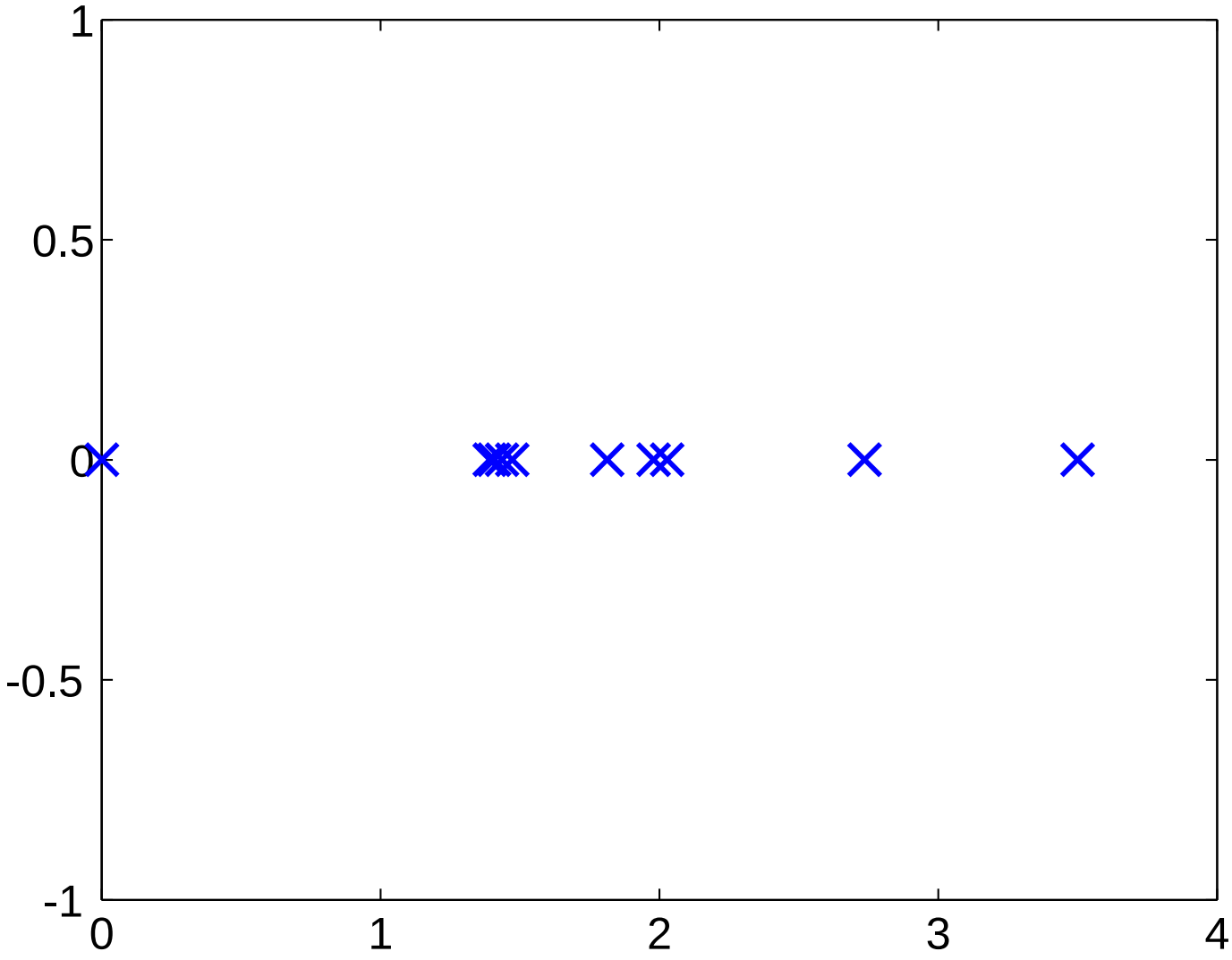
Convergence towards extreme eigenvalues goes faster if they are isolated (and the other eigenvalues are clustered).

Explanation (Heuristics). Again, consider the case where $0 < \lambda_1 < \lambda_2 < \ldots < \lambda_n$.

The shifted power method, with $\mathbf{A} - \sigma \mathbf{I}$ converges faster towards the eigenvector with eigenvalue $\lambda_1$ if the **spectral gap** $\frac{\lambda_2 - \lambda_1}{\lambda_n - \lambda_1}$ is larger. Because, then, with $\sigma \equiv (\lambda_1 + \lambda_n)/2$, the reduction factor $\frac{\lambda_2 - \sigma}{\lambda_n - \sigma}$ of the shifted power method is smaller.

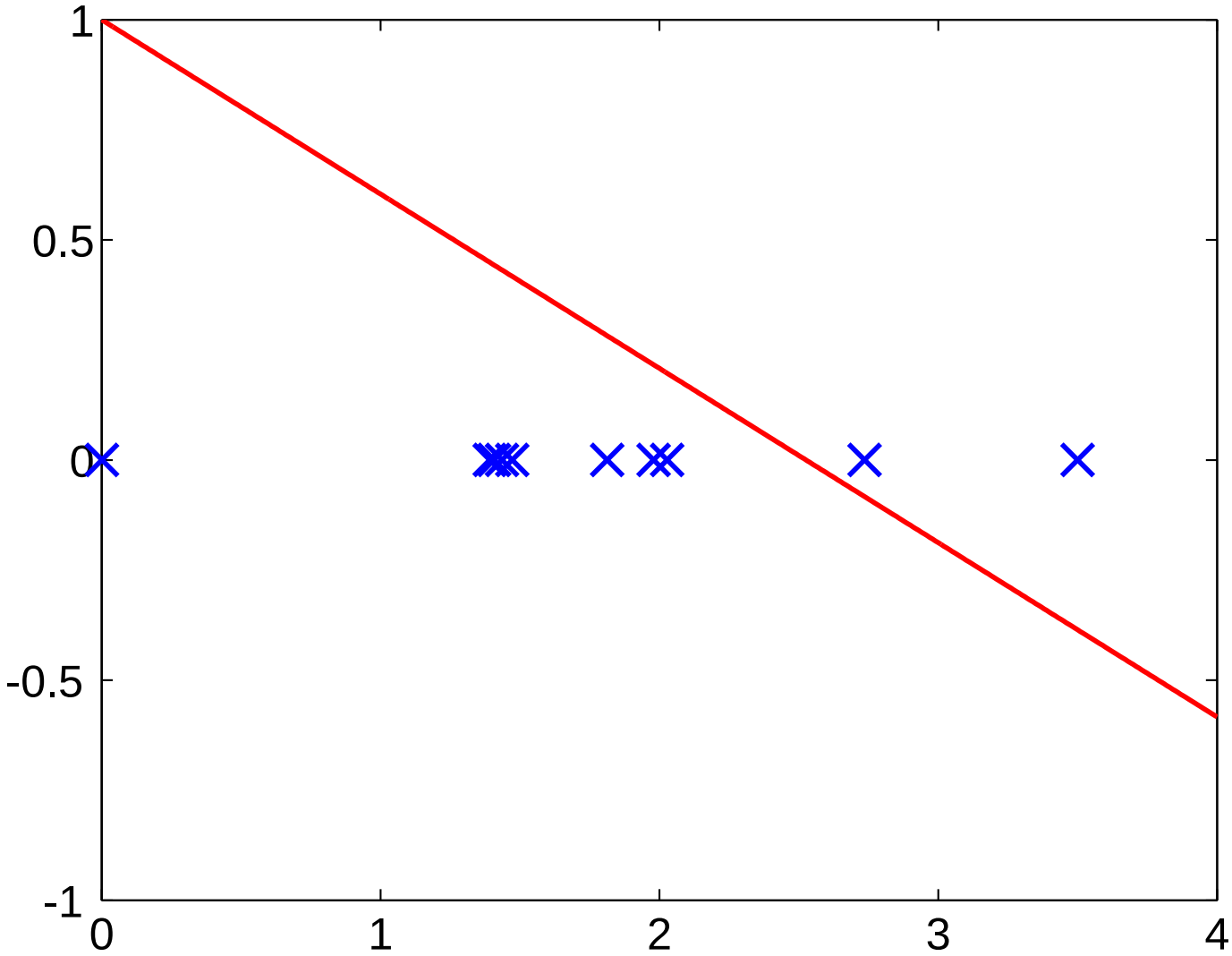
Since $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) = \mathcal{K}_k(\mathbf{A} - \sigma \mathbf{I}, \mathbf{r}_0)$, the Lanczos method will do better than the shifted power method: $\theta_1$ will converge faster towards $\lambda_1$ if the gap between $\lambda_1$ and the other eigenvalues is larger.

# CG convergence: Sparse Spectrum

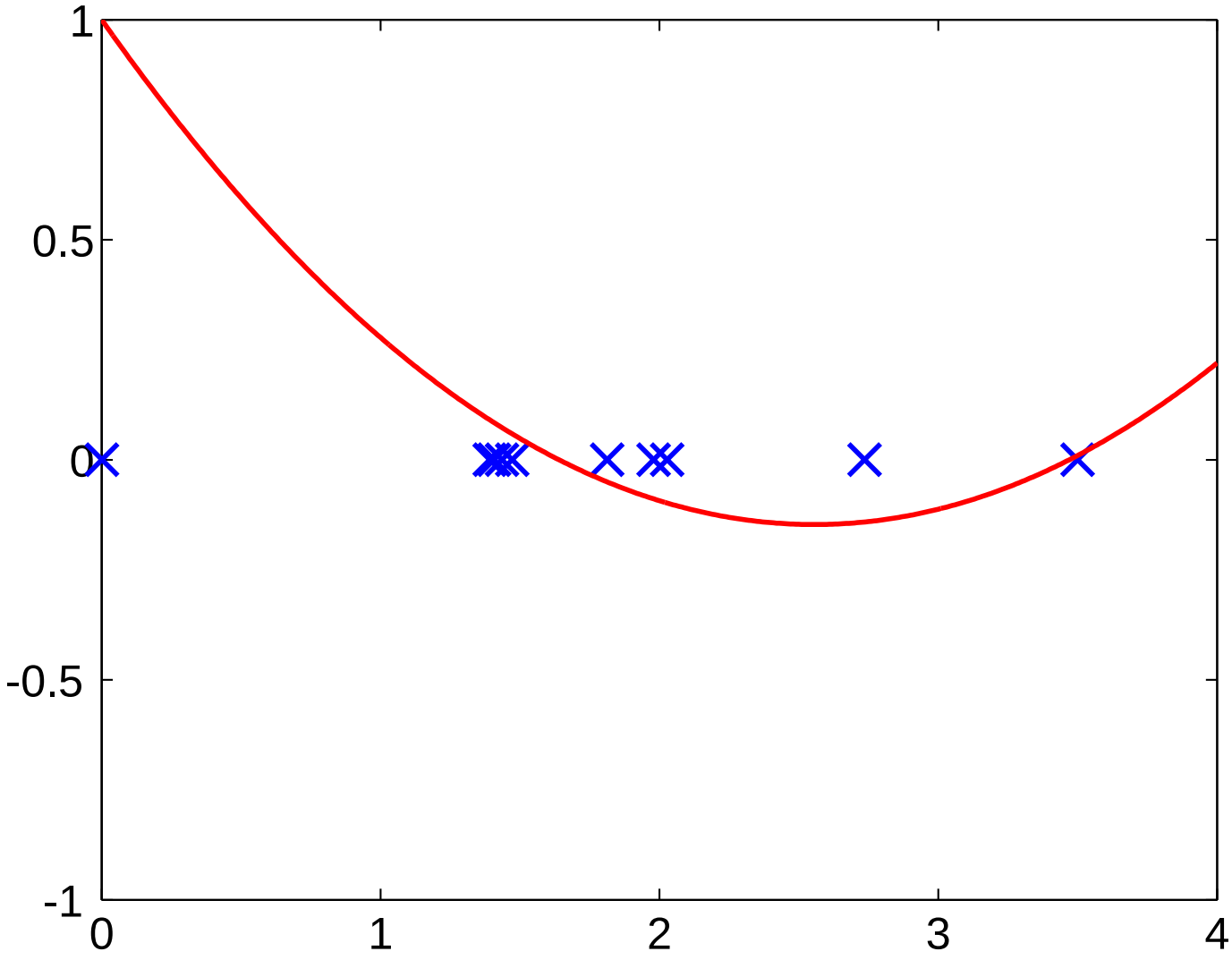



Spectrum of $A$

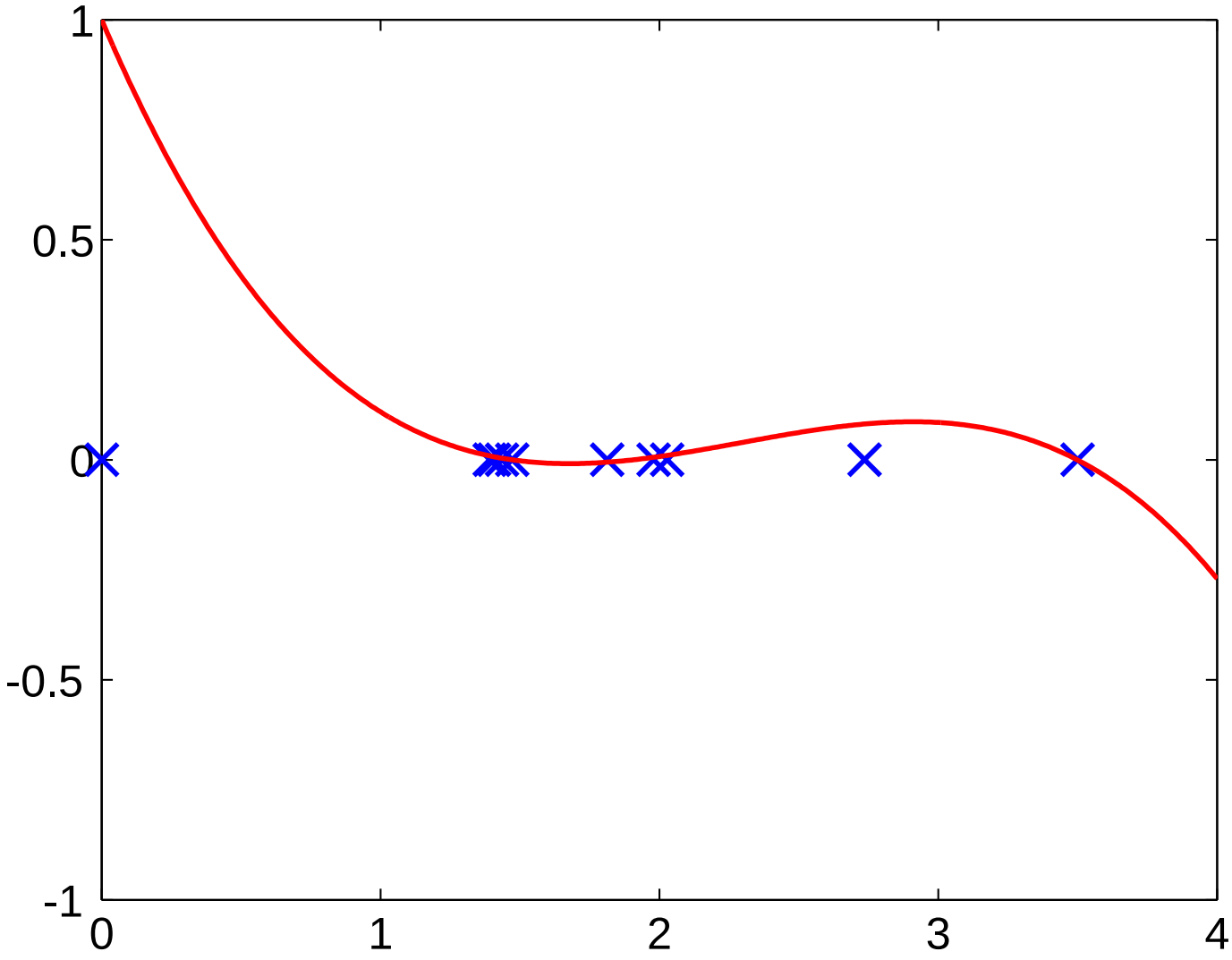
# CG convergence: Sparse Spectrum



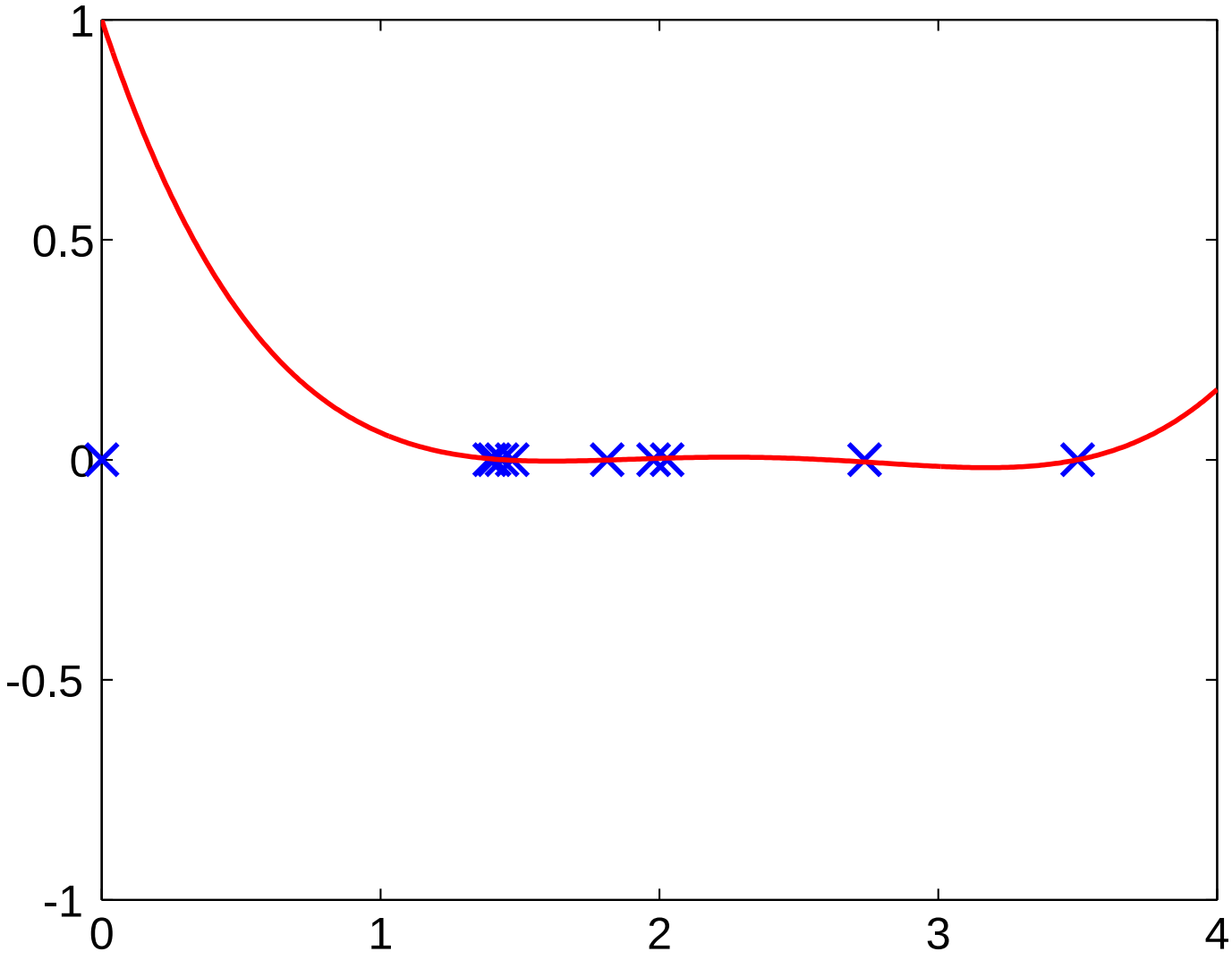
# CG convergence: Sparse Spectrum



# CG convergence: Sparse Spectrum

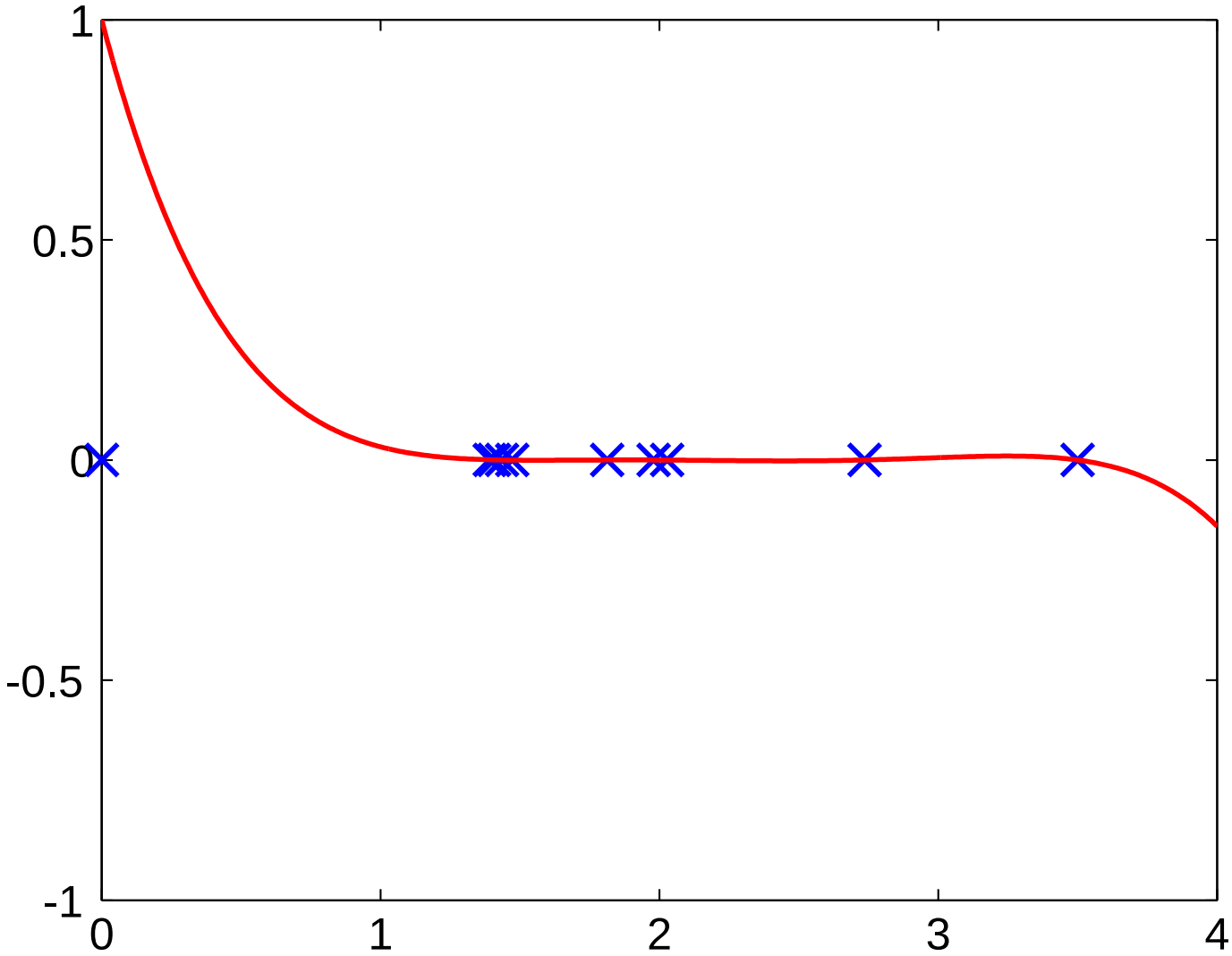


# CG convergence: Sparse Spectrum



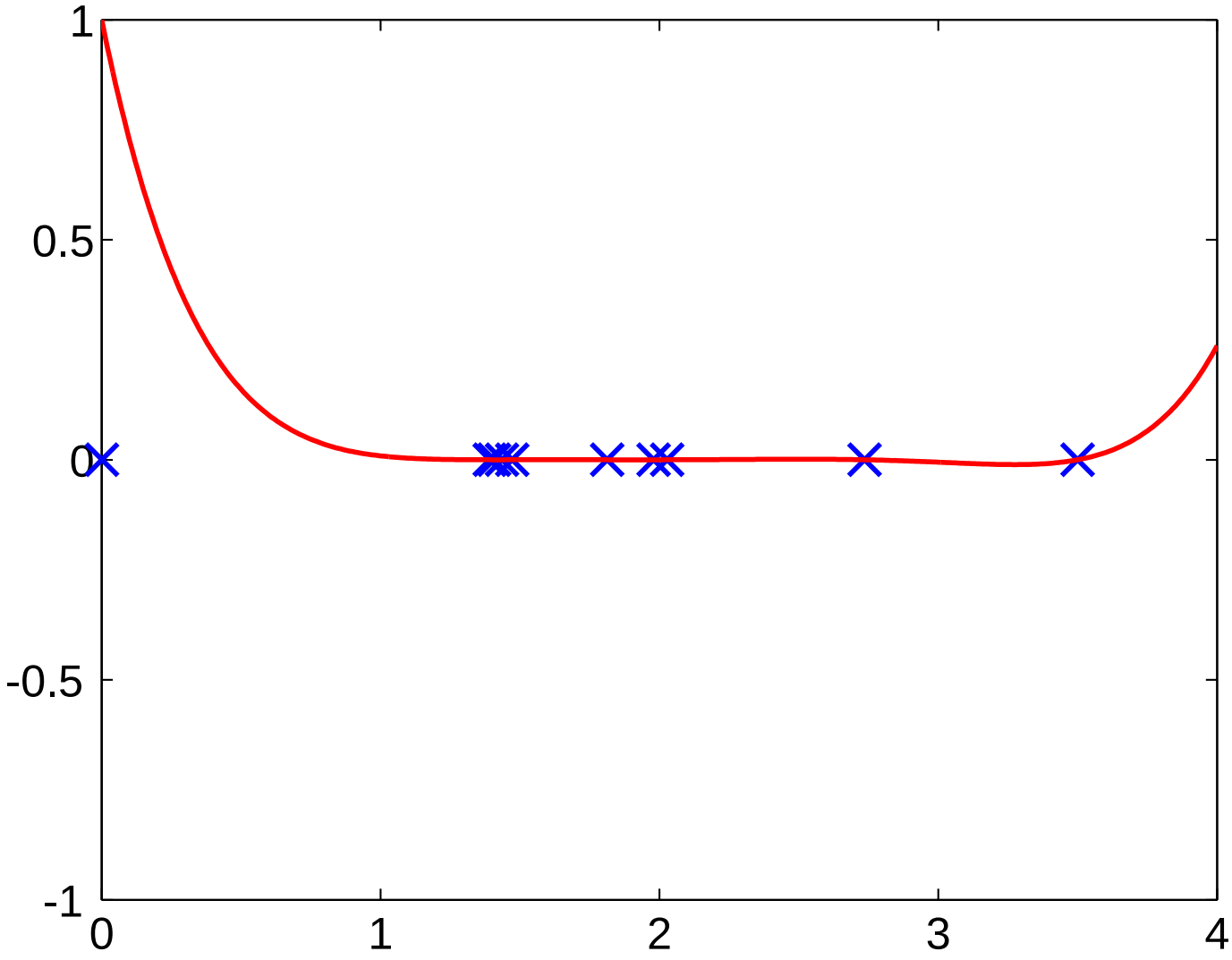

$p_4(\lambda)$

# CG convergence: Sparse Spectrum



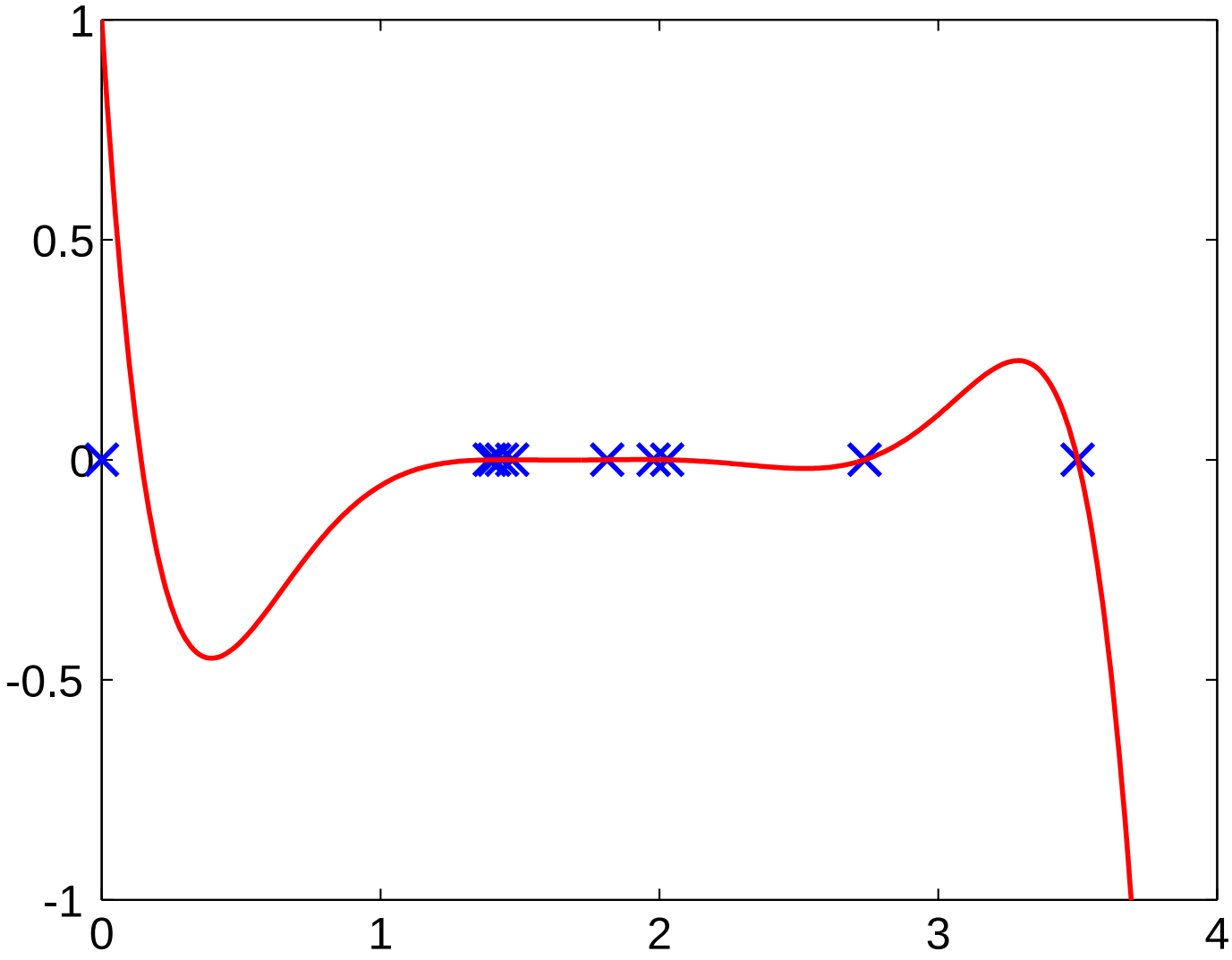

$p_5(\lambda)$

# CG convergence: Sparse Spectrum



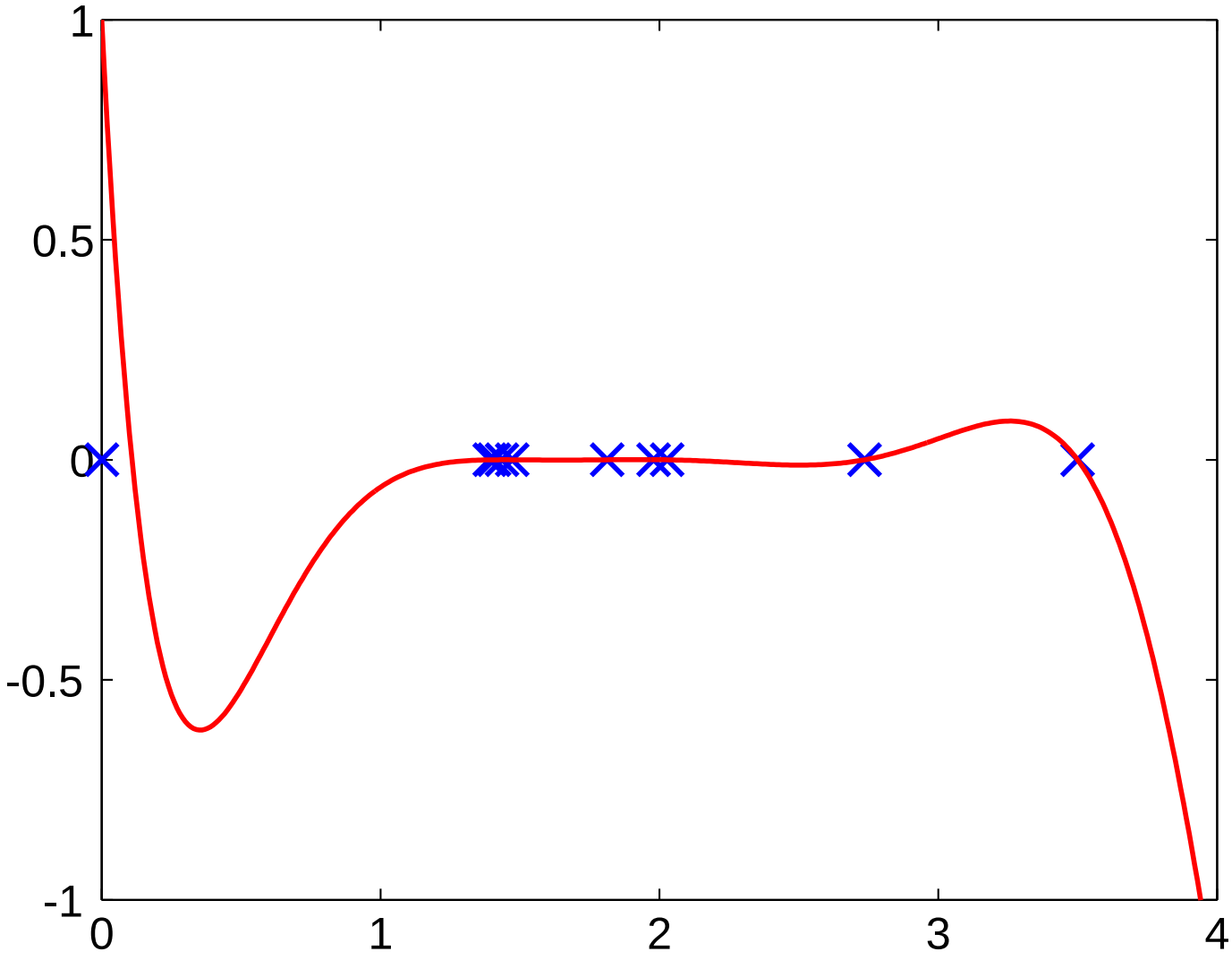

$p_6(\lambda)$

# CG convergence: Sparse Spectrum



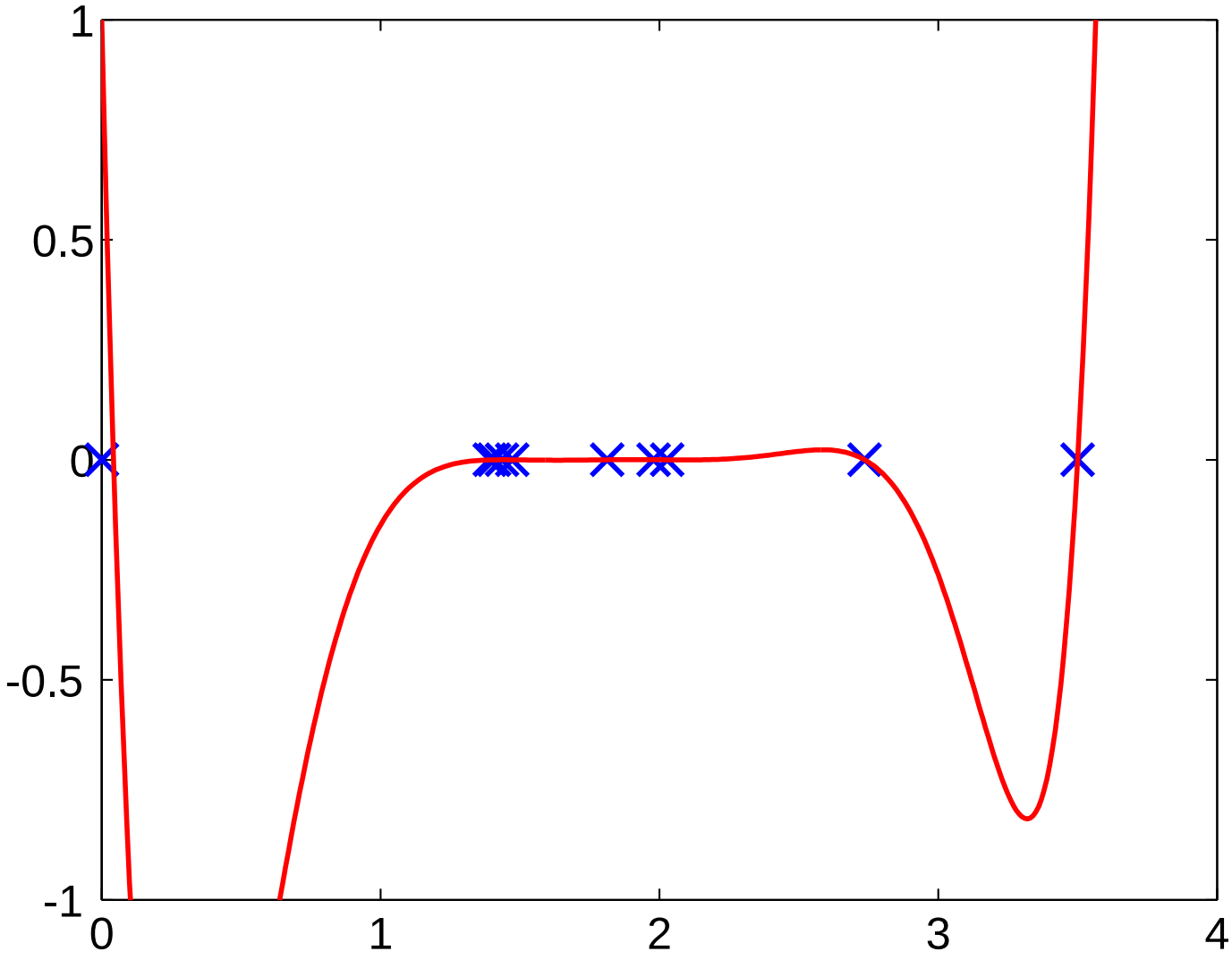

$p_7(\lambda)$

# CG convergence: Sparse Spectrum



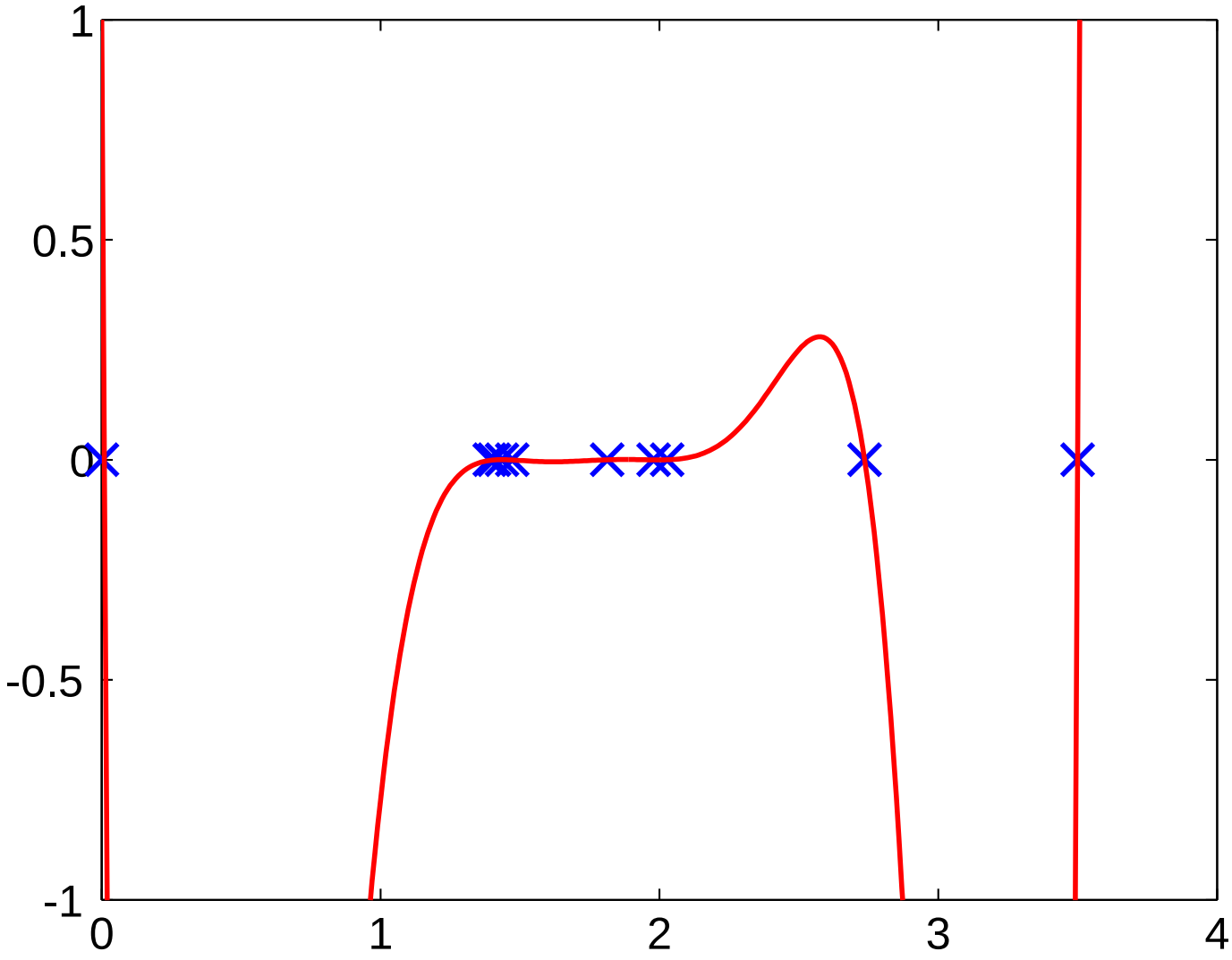

$p_8(\lambda)$

# CG convergence: Sparse Spectrum




$p_9(\lambda)$

# CG convergence: Sparse Spectrum



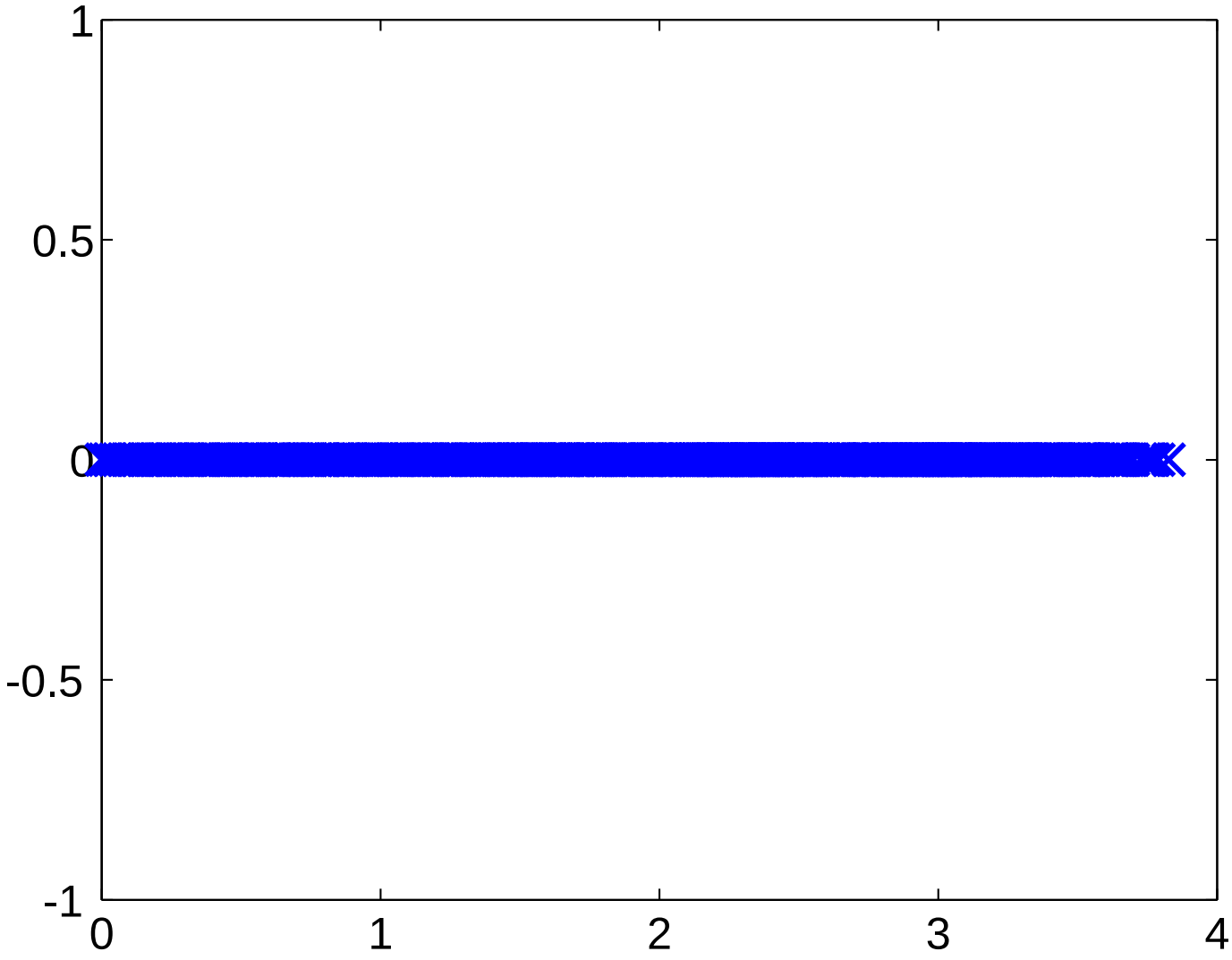

$p_{10}(\lambda)$

# CG convergence: small eigenvalues

Note that the speed of convergence towards an extreme eigenvalue also depends on the size of the component of the associated eigenvector in the initial $\mathbf{r}_0$.

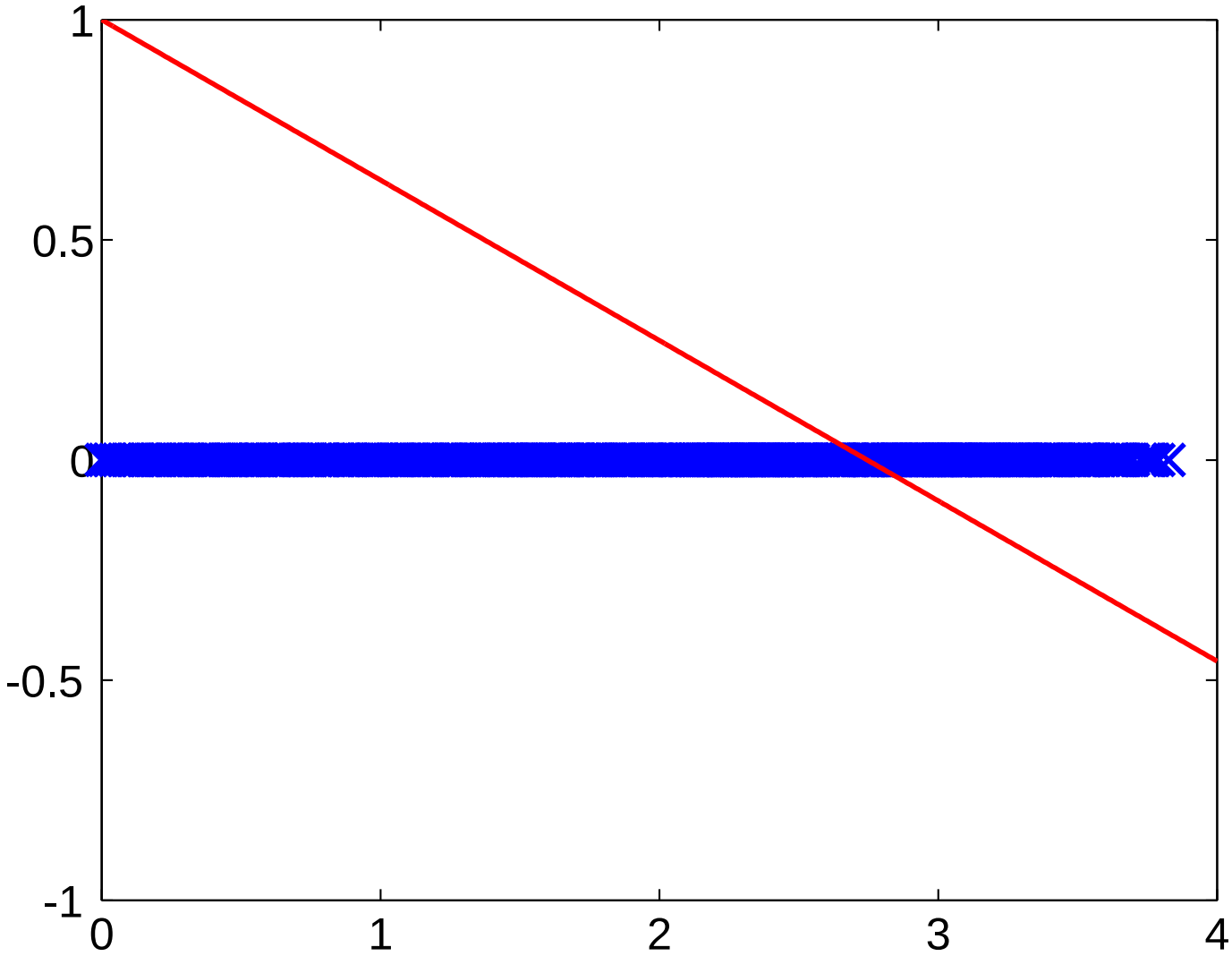
In the example, $\mathbf{b}$ was obtained as $\mathbf{b} = \mathbf{A}\mathbf{x}$ with $\mathbf{x} = \mathbf{1}$ and $\mathbf{A}$ diagonal. In practice, $\mathbf{x}$ often has moderate components also in the direction of eigenvectors with small eigenvalues. The small eigenvalues lead to small components of $\mathbf{r}_0 = \mathbf{b}$ (we took $\mathbf{x}_0 = \mathbf{0}$) in the direction of the associated eigenvectors. This explains the relative (as compared to the largest eigenvalue) slow convergence (in this example) towards the small eigenvalue (and is in line with the steepness of the polynomial near zero).

# CG convergence: Dense Spectrum

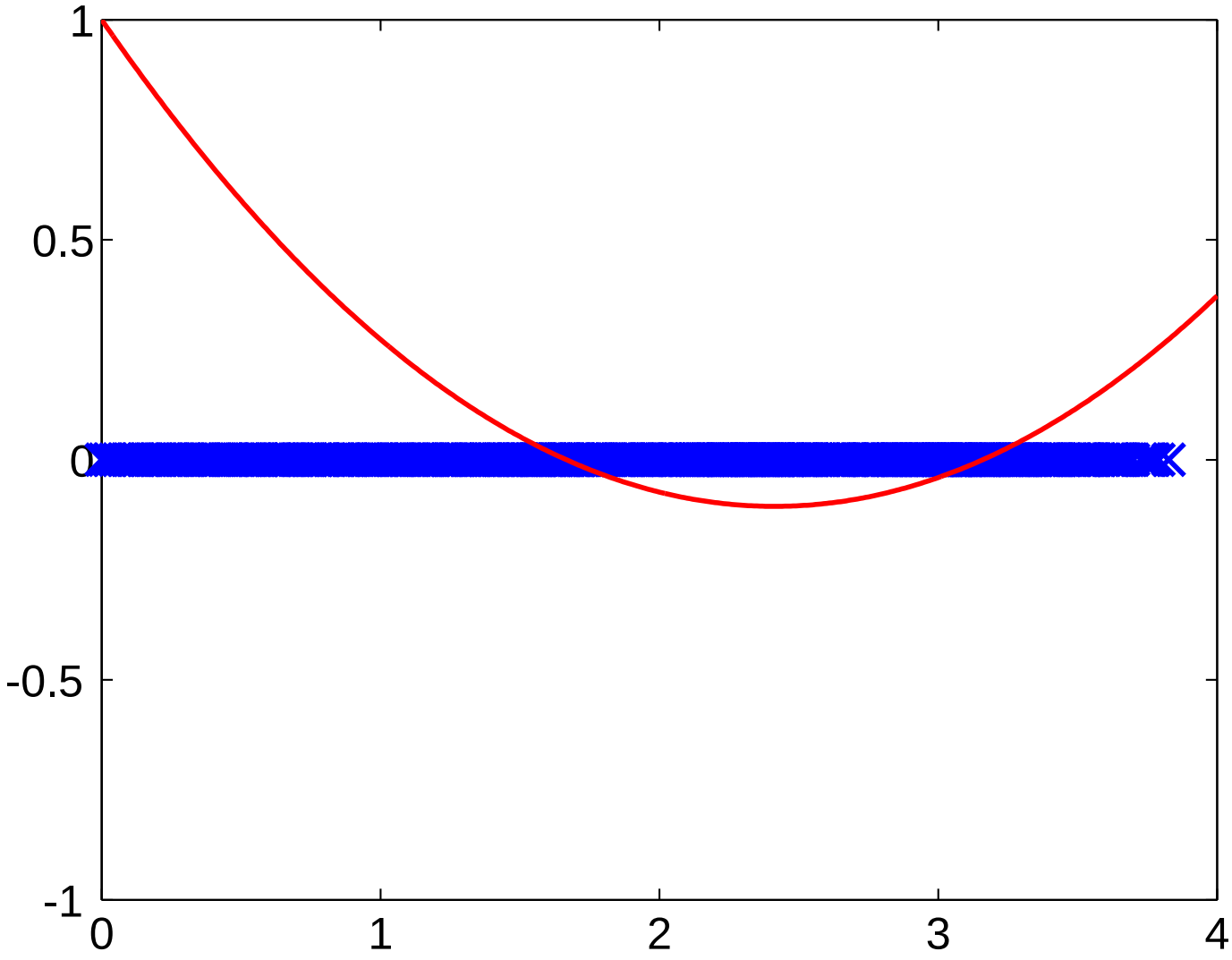



Spectrum of $A$

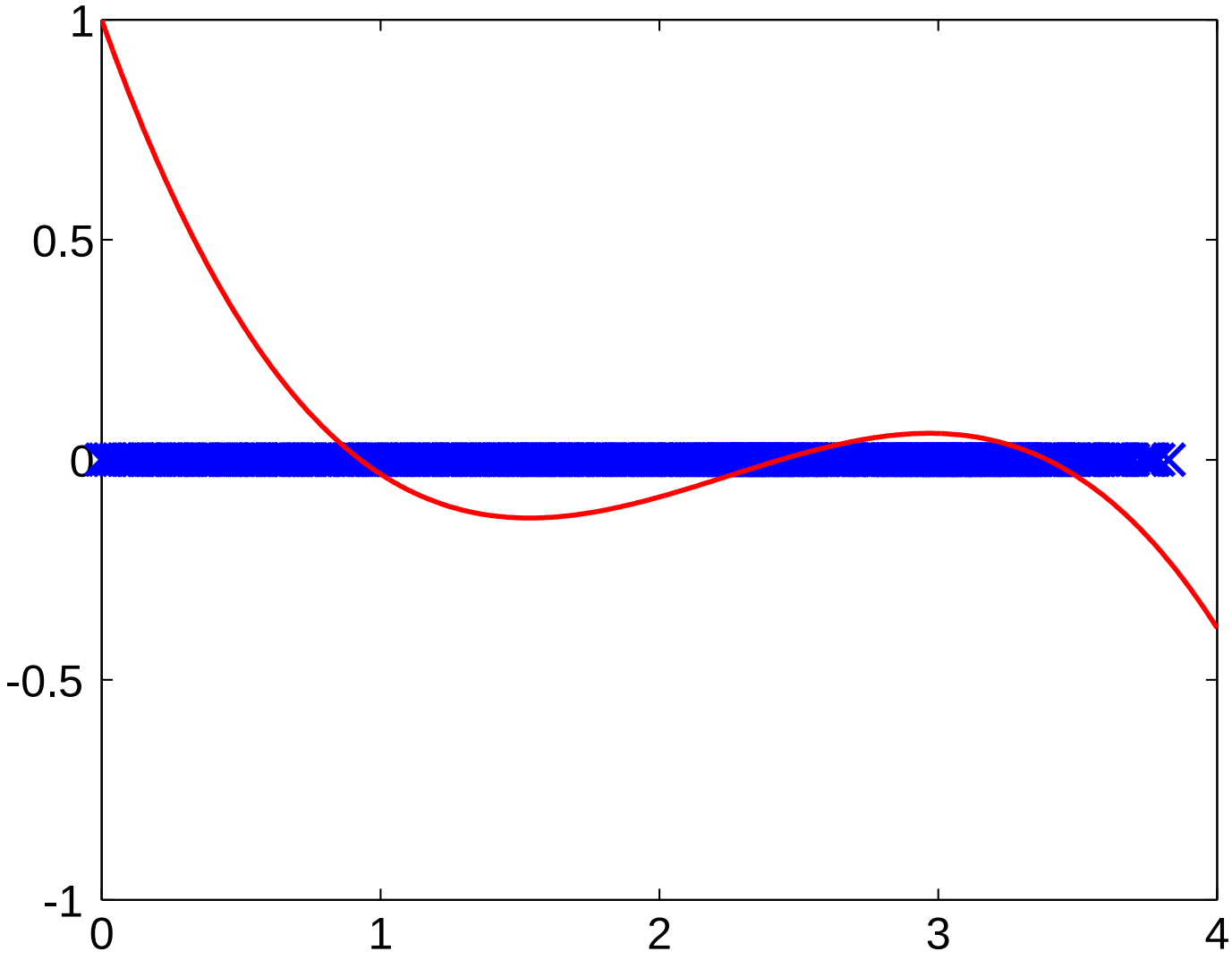
# CG convergence: Dense Spectrum



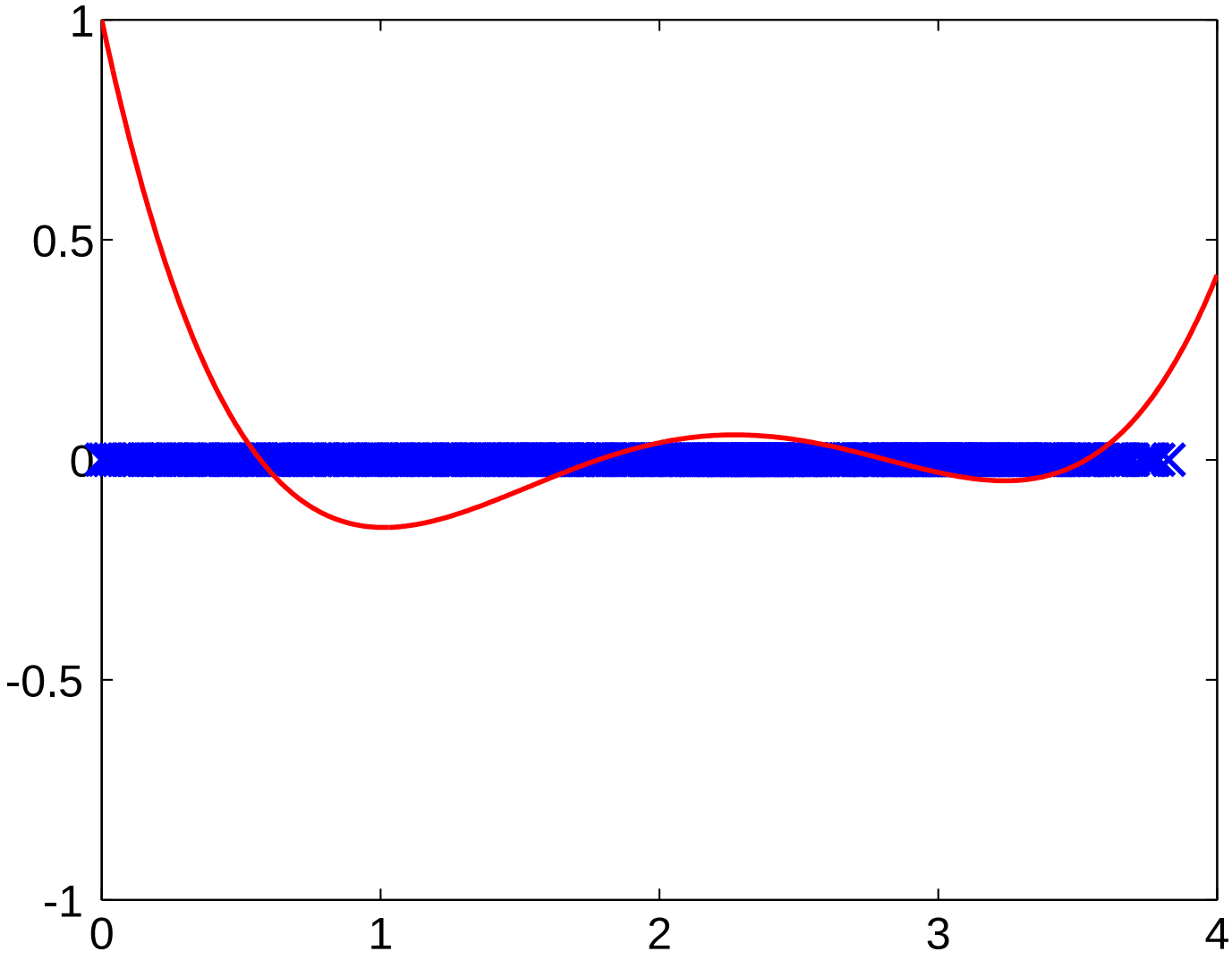
# CG convergence: Dense Spectrum



# CG convergence: Dense Spectrum

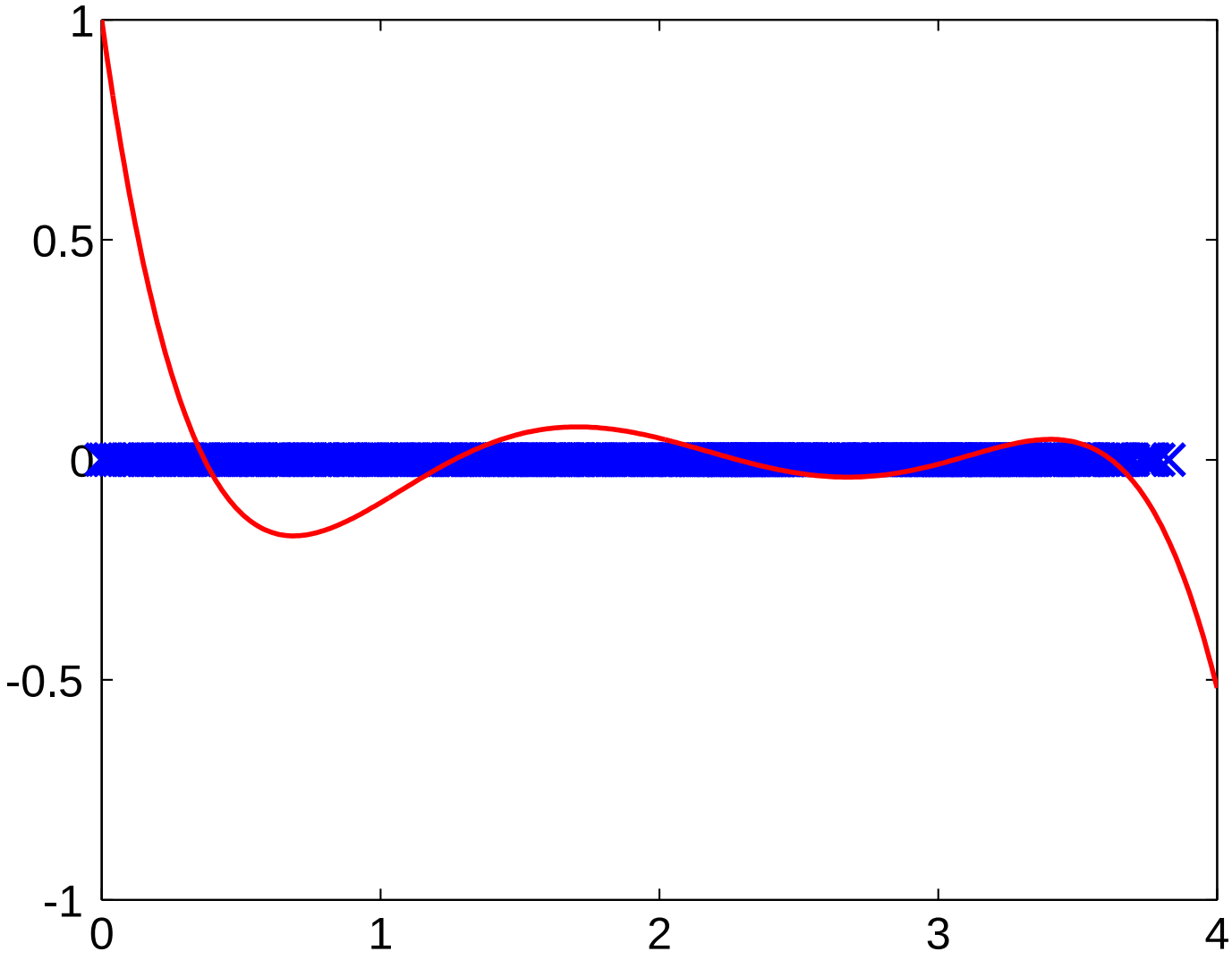


# CG convergence: Dense Spectrum



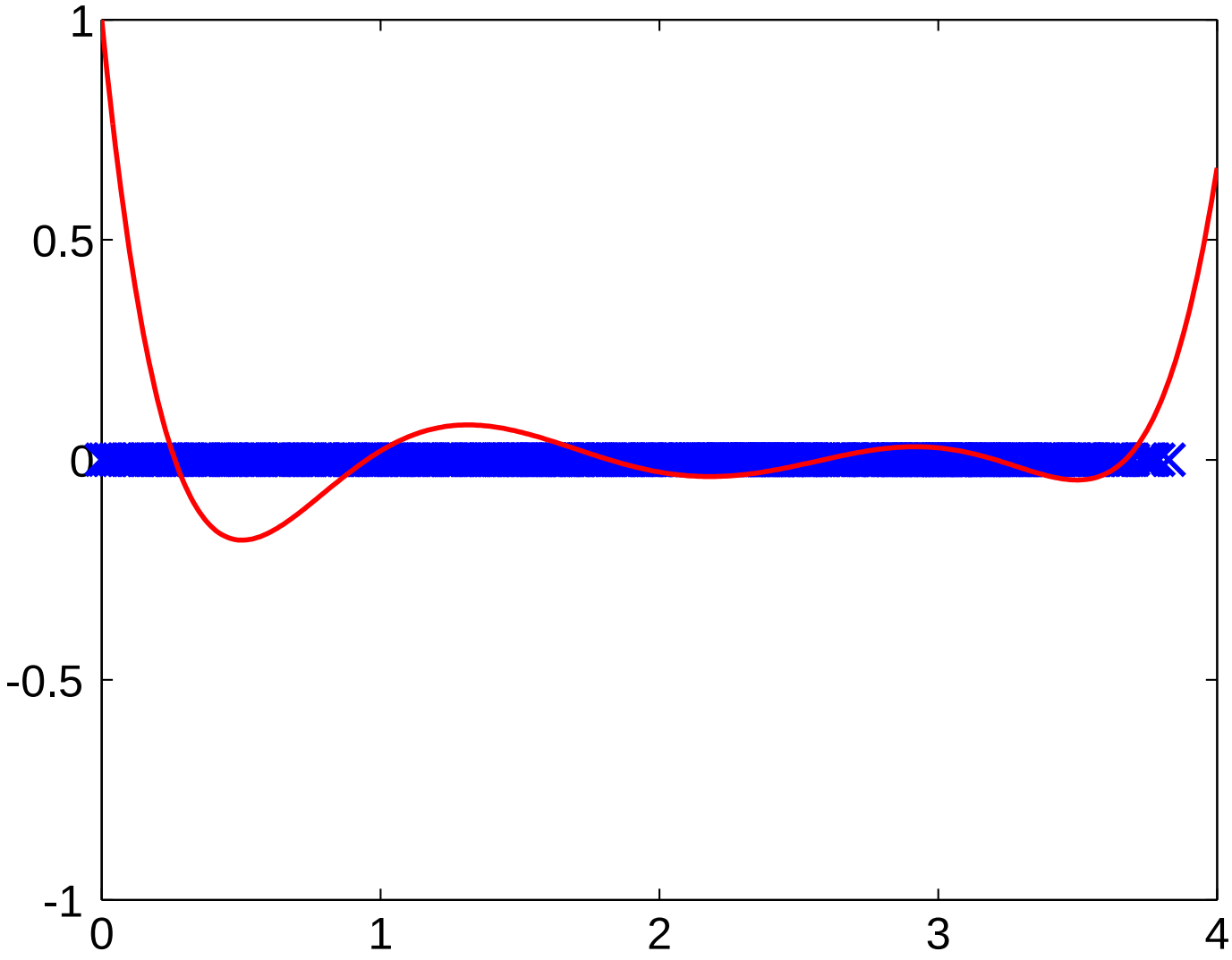

$p_4(\lambda)$

# CG convergence: Dense Spectrum



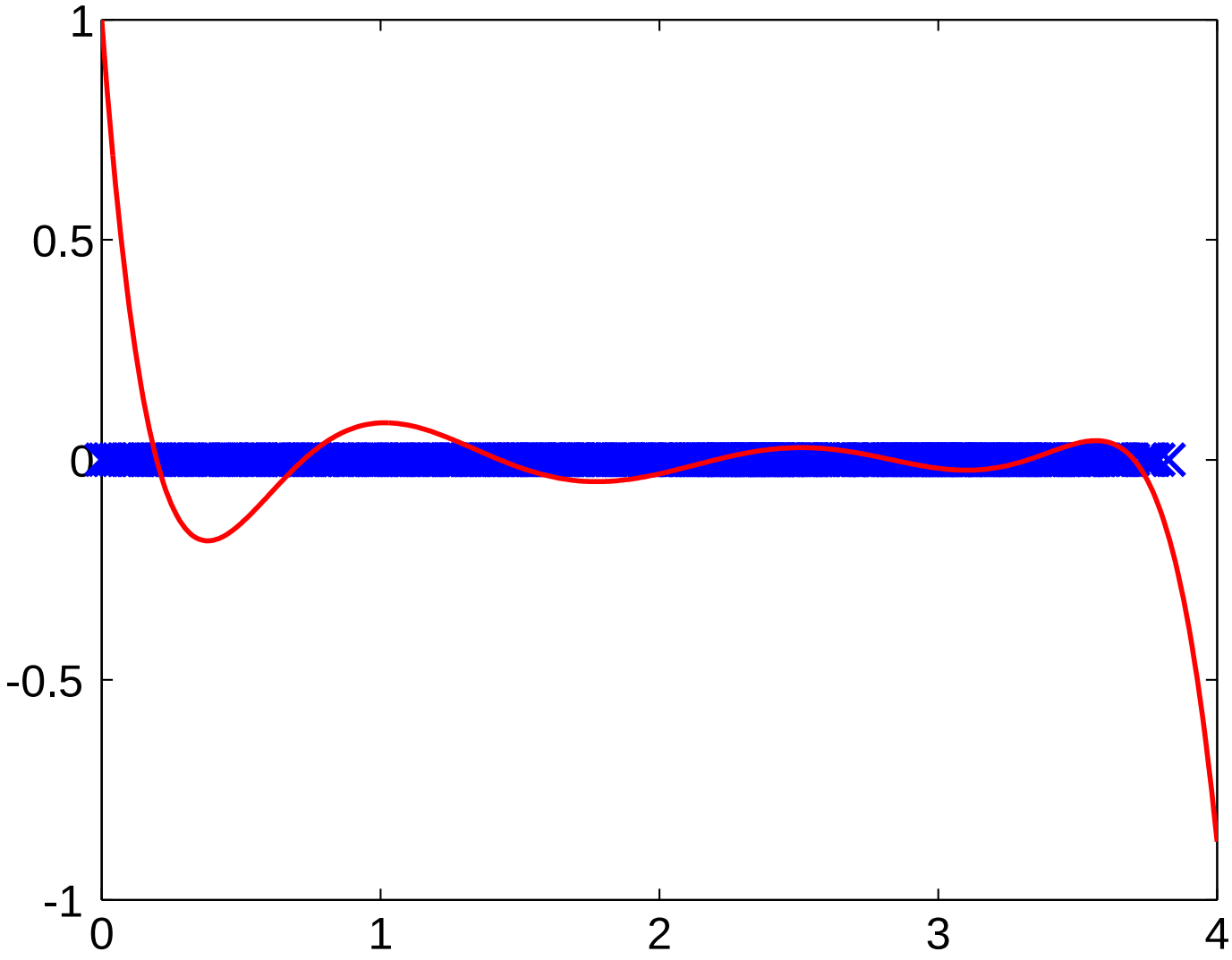

$p_5(\lambda)$

# CG convergence: Dense Spectrum



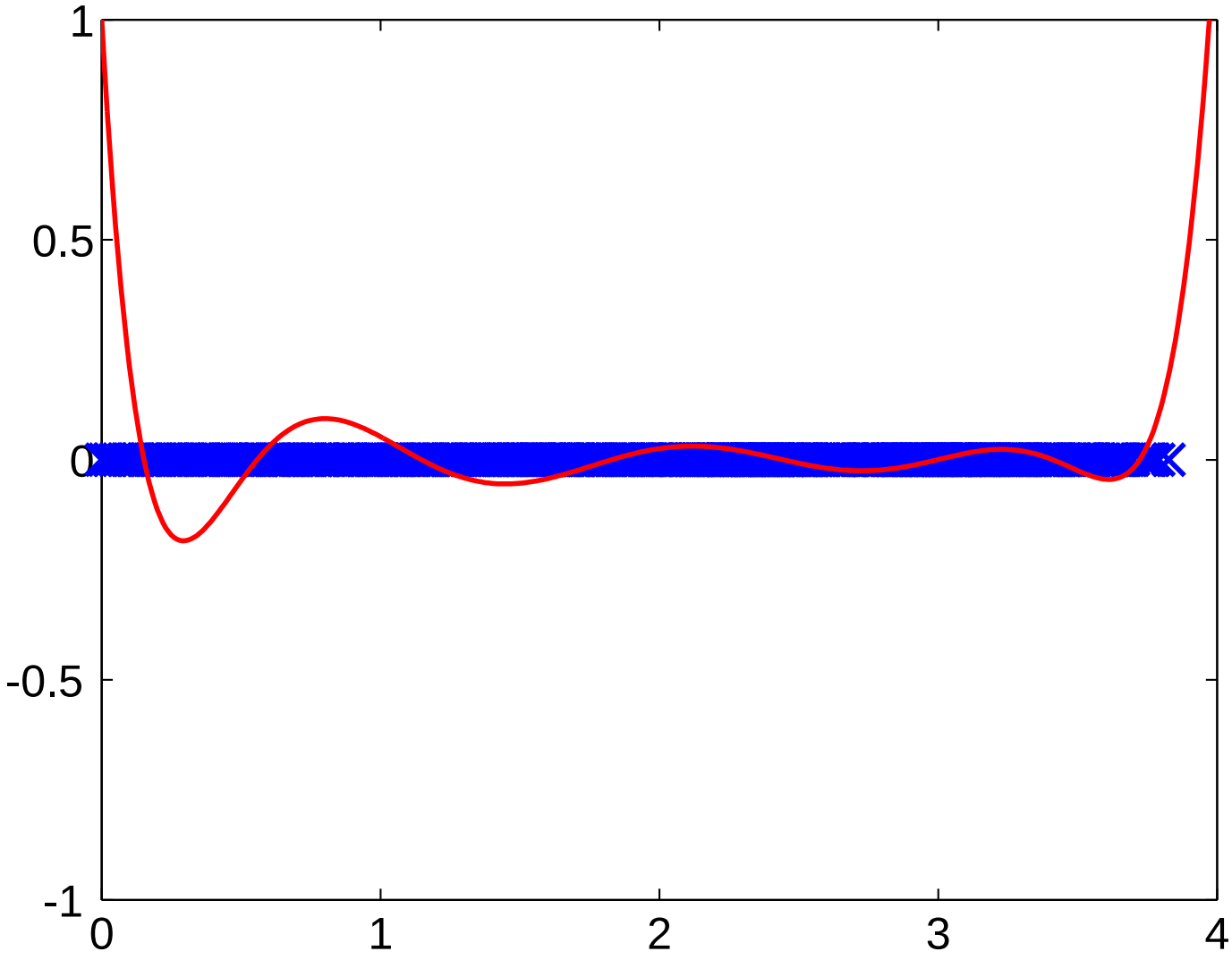

$p_6(\lambda)$

# CG convergence: Dense Spectrum



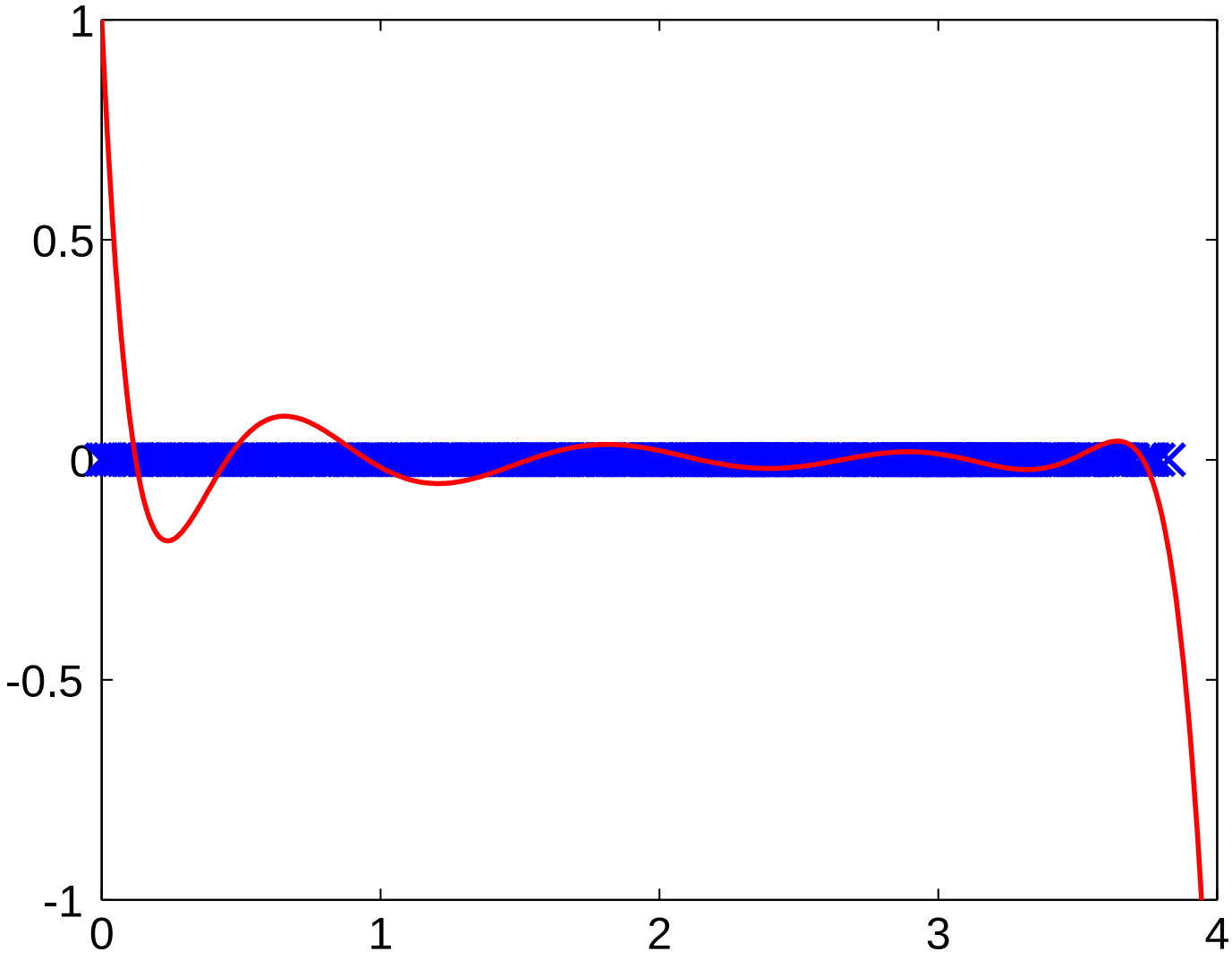

$p_7(\lambda)$

# CG convergence: Dense Spectrum



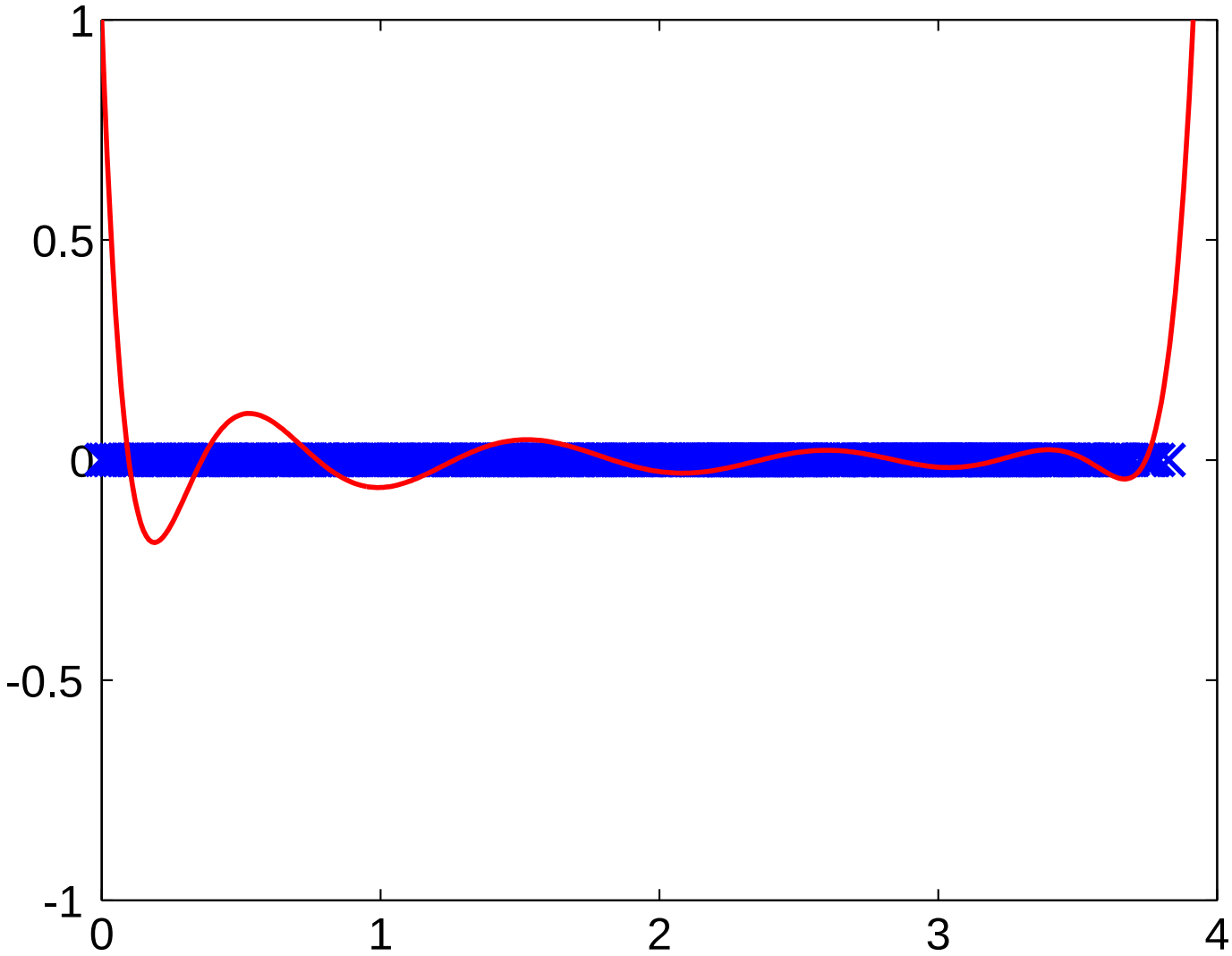

$p_8(\lambda)$

# CG convergence: Dense Spectrum




$p_9(\lambda)$

# CG convergence: Dense Spectrum




$p_{10}(\lambda)$

# Superlinear convergence (4)

Similar observations hold (with similar arguments) for *optimal* Krylov methods, as GMRES, FOM and GCR, for general matrices (though, for general matrices, the situation can be obscured by very skew eigenvectors, i.e., an ill-conditioned basis of eigenvectors).

# Minimising the residuals

CG minimises the $\mathbf{A}$-norm of the error. As we have seen before, another way to construct optimal approximations $\mathbf{x}_k$ is to minimise the residual, i.e. minimise

$$\|\mathbf{A}(\mathbf{x} - \mathbf{x}_k)\|_2 = \sqrt{\mathbf{r}_k^* \mathbf{r}_k}$$

over all $\mathbf{x}_k \in \mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$.

Before we solve this minimisation problem we recall the Lanczos relation.

# The Lanczos relation

Recall that, with

$$\underline{T}_k = \begin{bmatrix} \alpha_1 & \beta_2 & & & 0 \\ \beta_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_k \\ 0 & & & \beta_k & \alpha_k \\ & & & & \beta_{k+1} \end{bmatrix},$$

the Lanczos relation can be written as

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_{k+1}\underline{T}_k$$

# MINimal RESiduals

The problem is:

$$\text{find } \mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \vec{y}_k \text{ such that } \|\mathbf{r}_k\|_2 \text{ is minimal.}$$

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\,\mathbf{x}_k = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_k \vec{y}_k = \|\mathbf{r}_0\|_2\,\mathbf{v}_1 - \mathbf{A}\mathbf{V}_k \vec{y}_k.$$

Hence, minimise (as in GMRES)

$$\begin{aligned}
\|\mathbf{r}_k\|_2 &= \|\,\|\mathbf{r}_0\|_2\,\mathbf{v}_1 - \mathbf{A}\mathbf{V}_k \vec{y}_k\| \\
&= \|\,\|\mathbf{r}_0\|_2\,\mathbf{V}_{k+1}\,e_1 - \mathbf{V}_{k+1}\,\underline{T}_k\,\vec{y}_k\| \\
&= \|\mathbf{V}_{k+1}\,(\|\mathbf{r}_0\|_2\,e_1 - \underline{T}_k\,\vec{y}_k)\,\| \\
&= \|\,\|\mathbf{r}_0\|_2\,e_1 - \underline{T}_k\,\vec{y}_k\|
\end{aligned}$$

# MINRES (2)

Solving the small overdetermined system

$$\underline{T}_k \, \vec{y}_k = \|\mathbf{r}_0\|_2 \, e_1$$

provides iterates

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \, \vec{y}_k$$

that minimise the residual. The algorithm that results from solving the small system in least square sense using the QR-factorization $\underline{T}_k = \underline{Q}_k \, R_k$ of $\underline{T}_k$ is called **MINRES**:

$$\mathbf{x}_k = \mathbf{x}_0 + \left(\mathbf{V}_k \, R_k^{-1}\right) \left(\underline{Q}_k^*(\|\mathbf{r}_0\|_2 \, e_1)\right) .$$

The placing of the brackets, allows short recurrences.

# MINRES and GMRES

CG can be viewed as the Hermitian variant of FOM, while MINRES is the Hermitian variant of GMRES. For symmetric (HPD) matrices, CG is mathematically equivalent to FOM (i.e., in exact arithmetic, they have the same residuals in the same steps), MINRES is mathematically equivalent to GMRES.

Advantage. Fast convergence (smallest residuals).

Exploiting symmetry allows implementations with short recurrences: Lanczos combined with $(V_k U_k^{-1})(L_k^{-1} e_1)$ rather then with $V_k(U_k^{-1}(L_k^{-1} e_1))$.
Advantage. Highly efficient steps, low (fixed) on memory.

Disadvantage. More sensitive to perturbations.

# C onjugate R esiduals

CG has been designed for Hermitian positive definite matrices.

In a previous lecture we saw GCR for general matrices.
The Hermitian variant is CR and leads to minimal residuals also if $\mathbf{A}$ is Hermitian indefinite.
As GCR is mathematically equivalent to GMRES,
CR is mathematically equivalent to MINRES.

(G)CR can breakdown in the indefinite case, while MINRES is robust. MINRES is more (slightly?) expensive per step (six vector updates versus four for CR) and needs more memory.

# Conjugate Residuals (2)

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

$\mathbf{u}_{-1} = \mathbf{0},\ \mathbf{c}_{-1} = \mathbf{0},\ \rho_{-1} = 1$ % *Initialization*

for $k = 0, 1, \ldots,$ do

$\mathbf{u}_k = \mathbf{r}_k,\ \mathbf{c}_k = \mathbf{A}\mathbf{u}_k,$

$\rho_k = \mathbf{u}_k^* \mathbf{c}_k,\ \beta_k = \rho_k / \rho_{k-1}$

$\mathbf{u}_k \leftarrow \mathbf{u}_k + \beta_k\, \mathbf{u}_{k-1}$ % *Update direction vector*

$\mathbf{c}_k \leftarrow \mathbf{c}_k + \beta_k\, \mathbf{c}_{k-1}$ % *to avoid extra MVs*

$\sigma_k = \mathbf{c}_k^* \mathbf{c}_k,\ \alpha_k = \rho_k / \sigma_k$

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\, \mathbf{u}_k$ % *Update iterate*

$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k\, \mathbf{c}_k$ % *Update residual*

end for

# Conjugate Residuals (3)

Like CG, CR has many favourable properties:

- The method uses limited memory: only four vectors need to be stored;
- The method is optimal: the residual is minimised;
- The method is finite: the $n$th residual must be zero since it is optimal over the whole space;
- The method is robust if $\mathbf{A}$ is HPD, else $\rho_k = \mathbf{r}_k^* \mathbf{A} \mathbf{r}_k$ may be zero for some nonzero $\mathbf{r}_k$.

CR is less popular than CG since minimising the $\mathbf{A}$-norm of the error is often more natural. CG is also slightly cheaper.

# SYMM etric LQ

It is natural to try to minimise the *true* error $\mathbf{e}_k \equiv \mathbf{x} - \mathbf{x}_k$.
In the method **SYMMLQ** this is achieved by computing approximate solutions of the form

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{A}\mathbf{V}_k\,\vec{y}_k = \mathbf{x}_0 + \mathbf{V}_{k+1}\,\underline{T}_k\,\vec{y}_k$$

($\mathbf{A}\mathbf{V}_k\,\vec{y}_k$ rather than $\mathbf{V}_k\,\vec{y}_k$ as before) and minimising

$$\|\mathbf{x} - \mathbf{x}_k\|_2 = \|\mathbf{x} - \mathbf{x}_0 - \mathbf{A}\mathbf{V}_k\,\vec{y}_k\|_2$$

with repect to $\vec{y}_k$, or, equivalently, solving

$$(\mathbf{A}\mathbf{V}_k)^*(\mathbf{e}_0 - \mathbf{A}\mathbf{V}_k\,\vec{y}_k) = 0 \;\Leftrightarrow\; (\mathbf{A}\mathbf{V}_k)^*\mathbf{A}\mathbf{V}_k\,\vec{y}_k = \mathbf{V}_k^*\mathbf{A}\mathbf{e}_0 = \|\mathbf{r}_0\|_2\,e_1.$$

# SYMMLQ (2)

Using the Lanczos relation we get

$$(\mathbf{A}\mathbf{V}_k)^*\mathbf{A}\mathbf{V}_k\ \vec{y}_k = (\mathbf{V}_{k+1}\,\underline{T}_k)^*(\mathbf{V}_{k+1}\,\underline{T}_k)\vec{y}_k = \|\mathbf{r}_0\|_2\, e_1$$

or

$$\underline{T}_k^*\,(\underline{T}_k\,\vec{y}_k) = \|\mathbf{r}_0\|_2\, e_1.$$

This problem is solved with the QR-factorisation of $\underline{T}_k$, yielding

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_{k+1}\,\underline{T}_k\,\vec{y}_k = \mathbf{x}_0 + \left(\mathbf{V}_{k+1}\,\underline{Q}_k\right)\left(R_k^{*\,-1}(\|\mathbf{r}_0\|_2\, e_1)\right).$$

SYMMLQ is a stable method for solving symmetric *indefinite* linear systems.

# SYMMLQ (3): the naming

If $\vec{y}_k$ solves $\underline{T}_k^* (\underline{T}_k \vec{y}_k) = \|\mathbf{r}_0\|_2 \, e_1$, then $\vec{z}_{k+1} \equiv \underline{T}_k \vec{y}_k$ solves

$$\underline{T}_k^* \vec{z}_{k+1} = \|\mathbf{r}_0\|_2 \, e_1 \quad \text{in the } \textbf{least norm} \text{ sense,}$$

i.e., among all solutions, $\underline{T}_k \vec{y}_k$ is the one with smallest 2-norm.
This problem is solved with the LQ-factorization of $\underline{T}_k^*$
(which is obtained by $\cdot^*$ the QR-factorisation of $\underline{T}_k$):

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_{k+1} \vec{z}_{k+1} \quad \text{with} \quad \vec{z}_{k+1} = \underline{Q}_k (R_k^{*-1} (\|\mathbf{r}_0\|_2 \, e_1)).$$

This explains the naming for this method.

In MINRES $\vec{y}_k$ is the **least square** solution of $\underline{T}_k \vec{y}_k = \|\mathbf{r}_0\|_2 \, e_1$:

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \vec{y}_k \quad \text{with} \quad \vec{y}_k = R_k^{-1} (\underline{Q}_k^* (\|\mathbf{r}_0\|_2 \, e_1)).$$

# Concluding remarks

Today, we discussed Krylov methods for symmetric systems. These methods combine an optimal error reduction with short recurrences, and hence limited memory requirements.

Last week we discussed GMRES, which solves nonsymmetric problems while minimising the norm of the residual over the Krylov subspace. For this you need to store all the basis vectors.

In the next lessons we will investigate methods for nonsymmetric systems that only require a limited number of vectors, similar to the methods we discussed today. However, as we will see, these method do *not* minimise an error norm.