

PROJECTION METHODS

Projection Methods

- The main idea of projection methods is to extract an approximate solution from a subspace.
- We define a subspace of approximants of dimension m and a set of m conditions to extract the solution
- These conditions are typically expressed by orthogonality constraints.
- This defines one basic step which is repeated until convergence (alternatively the dimension of the subspace is increased until convergence).

Example:

Each relaxation step in Gauss-Seidel can be viewed as a projection step

Background on projectors

➤ A projector is a linear operator that is idempotent:

$$P^2 = P$$

A few properties:

- P is a projector iff $I - P$ is a projector
- $x \in \text{Ran}(P)$ iff $x = Px$ iff $x \in \text{Null}(I - P)$
- This means that : $\text{Ran}(P) = \text{Null}(I - P)$.
- Any $x \in \mathbb{R}^n$ can be written (uniquely) as $x = x_1 + x_2$,
 $x_1 = Px \in \text{Ran}(P)$ $x_2 = (I - P)x \in \text{Null}(P)$ - So:

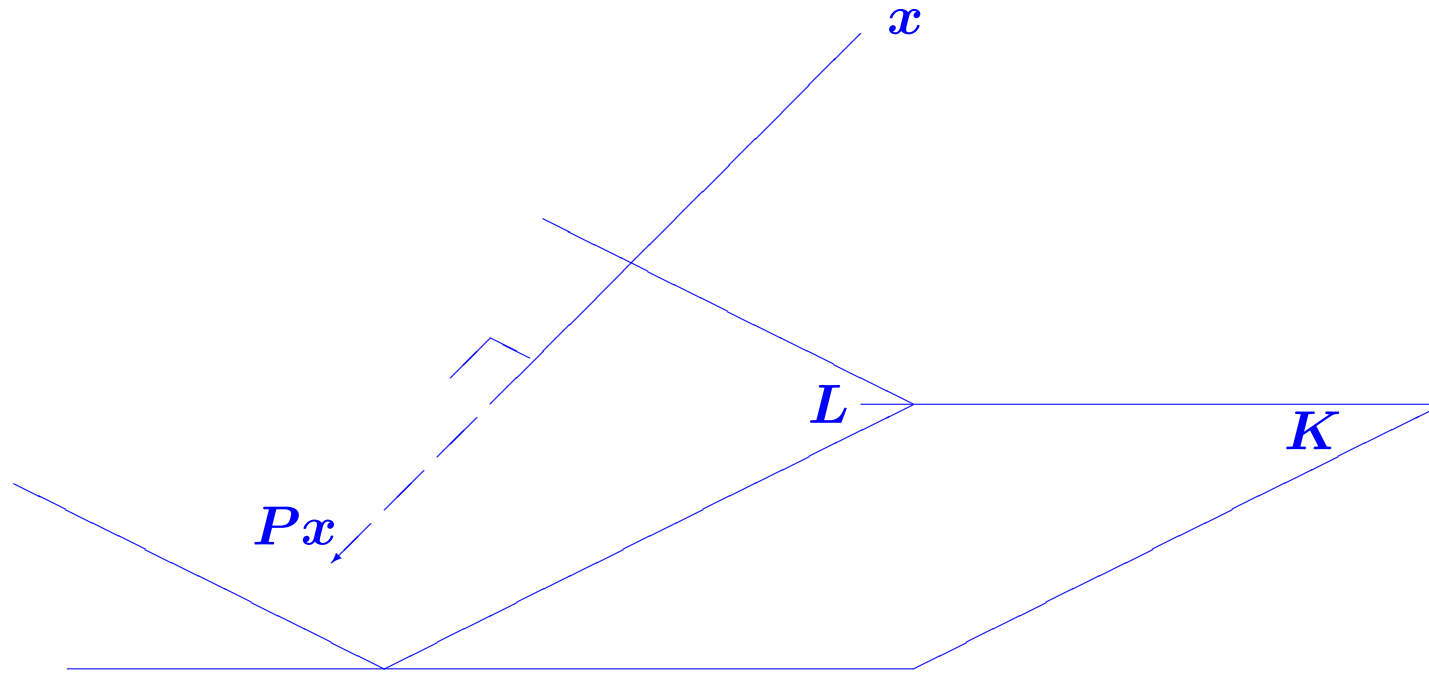
$$\mathbb{R}^n = \text{Ran}(P) \oplus \text{Null}(P)$$

Background on projectors (Continued)

Decomposition $\mathbb{R}^n = K \oplus S$ defines a (unique) projector P :

- From $x = x_1 + x_2$, set $Px = x_1$.
- For this P : $\text{Ran}(P) = K$ and $\text{Null}(P) = S$.
- Note: $\dim(K) = m$, $\dim(S) = n - m$.
- Pb: express mapping $x \rightarrow u = Px$ in terms of K, S
- Note $u \in K, x - u \in S$
- Express 2nd part with m constraints: let $L = S^\perp$, then
 - Projection onto K and orthogonally to L

$$u = Px \text{ iff } \begin{cases} u \in K \\ x - u \perp L \end{cases}$$



- Illustration: P projects onto K and orthogonally to L
- When $L = K$ projector is orthogonal.
- Note: $Px = 0$ iff $x \perp L$.

Projection methods for linear systems

➤ Initial Problem:

$$b - Ax = 0$$

➤ Given two subspaces K and L of \mathbb{R}^N of dimension m , define ...

Approximate problem:

Find $\tilde{x} \in K$ such that $\underbrace{b - A\tilde{x}} \perp L$
Petrov-Galerkin cond.

➤ m degrees of freedom (K) + m constraints (L) \rightarrow

➤ To solve: A small linear system ('projected problem')

➤ Basic projection step. Typically a sequence of such steps are applied

➤ With a nonzero initial guess x_0 , approximate problem is

$$\text{Find } \tilde{x} \in x_0 + K \text{ such that } b - A\tilde{x} \perp L$$

Write $\tilde{x} = x_0 + \delta$ and $r_0 = b - Ax_0$. \rightarrow system for δ :

$$\text{Find } \delta \in K \text{ such that } r_0 - A\delta \perp L$$

 Formulate Gauss-Seidel as a projection method -

 Generalize Gauss-Seidel by defining subspaces consisting of 'blocks' of coordinates $\text{span}\{e_i, e_{i+1}, \dots, e_{i+p}\}$

Matrix representation:

Let

- $V = [v_1, \dots, v_m]$ a basis of K &
- $W = [w_1, \dots, w_m]$ a basis of L

➤ Write approximate solution as $\tilde{x} = x_0 + \delta \equiv x_0 + Vy$ where $y \in \mathbb{R}^m$.
Then Petrov-Galerkin condition yields:

$$W^T(r_0 - AVy) = 0$$

➤ Therefore,

$$\tilde{x} = x_0 + V[W^T AV]^{-1}W^T r_0$$

Remark: In practice $W^T AV$ is known from algorithm and has a simple structure [tridiagonal, Hessenberg,..]

Prototype Projection Method

Until Convergence Do:

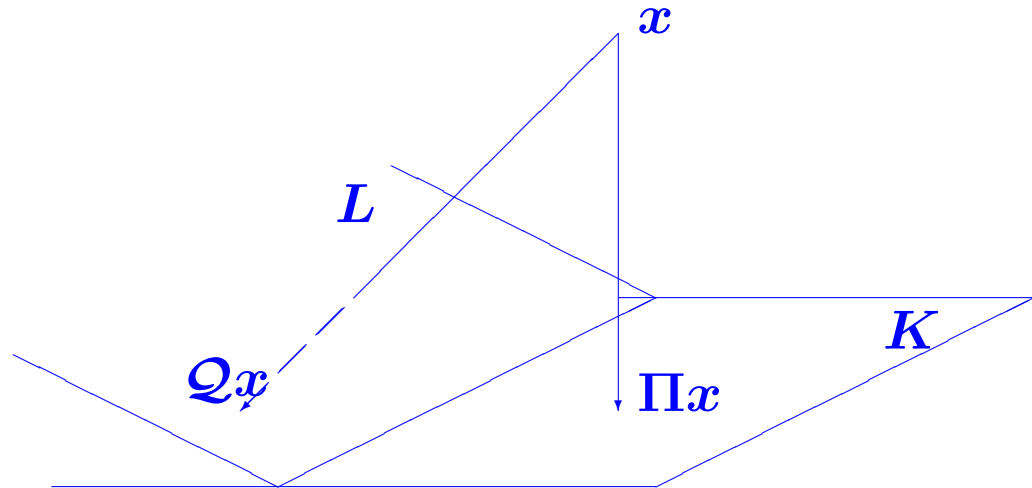
1. Select a pair of subspaces K , and L ;
2. Choose bases:
 $V = [v_1, \dots, v_m]$ for K and
 $W = [w_1, \dots, w_m]$ for L .

3. Compute :
 $r \leftarrow b - Ax,$
 $y \leftarrow (W^T AV)^{-1} W^T r,$
 $x \leftarrow x + Vy.$

Projection methods: Operator form representation

- Let Π = the orthogonal projector onto K and \mathcal{Q} the (oblique) projector onto K and orthogonally to L .

$$\begin{array}{l} \Pi x \in K, \quad x - \Pi x \perp K \\ \mathcal{Q}x \in K, \quad x - \mathcal{Q}x \perp L \end{array}$$



Assumption: no vector of K is \perp to L

In the case $x_0 = 0$, approximate problem amounts to solving

$$Q(b - Ax) = 0, \quad x \in K$$

or in operator form (solution is Πx)

$$Q(b - A\Pi x) = 0$$

Question: what accuracy can one expect?

➤ Let x^* be the exact solution. Then

1) We cannot get better accuracy than $\|(I - \Pi)x^*\|_2$, i.e.,

$$\|\tilde{x} - x^*\|_2 \geq \|(I - \Pi)x^*\|_2$$

2) The residual of the exact solution for the approximate problem satisfies:

$$\|b - QA\Pi x^*\|_2 \leq \|QA(I - \Pi)\|_2 \|(I - \Pi)x^*\|_2$$

Two Important Particular Cases.

1. $L = K$

- When A is SPD then $\|x^* - \tilde{x}\|_A = \min_{z \in K} \|x^* - z\|_A$.
- Class of Galerkin or Orthogonal projection methods
- Important member of this class: Conjugate Gradient (CG) method

2. $L = AK$

In this case $\|b - A\tilde{x}\|_2 = \min_{z \in K} \|b - Az\|_2$

- Class of Minimal Residual Methods: CR, GCR, ORTHOMIN, GMRES, CGNR, ...

One-dimensional projection processes

$$\begin{aligned} K &= \text{span}\{d\} \\ &\text{and} \\ L &= \text{span}\{e\} \end{aligned}$$

Then $\tilde{x} = x + \alpha d$. Condition $r - A\delta \perp e$ yields

$$\alpha = \frac{(r, e)}{(Ad, e)}$$

➤ Three popular choices:

(1) Steepest descent

(2) Minimal residual iteration

(3) Residual norm steepest descent

1. Steepest descent.

A is SPD. Take at each step $d = r$ and $e = r$.

Iteration:

$$\begin{aligned} r &\leftarrow b - Ax, \\ \alpha &\leftarrow (r, r) / (Ar, r) \\ x &\leftarrow x + \alpha r \end{aligned}$$

➤ Each step minimizes $f(x) = \|x - x^*\|_A^2 = (A(x - x^*), (x - x^*))$ in direction $-\nabla f$.

➤ Convergence guaranteed if A is SPD.

 As is formulated, the above algorithm requires 2 ‘matvecs’ per step. Reformulate it so only one is needed.

Convergence based on the Kantorovitch inequality: Let B be an SPD matrix, λ_{max} , λ_{min} its largest and smallest eigenvalues. Then,

$$\frac{(Bx, x)(B^{-1}x, x)}{(x, x)^2} \leq \frac{(\lambda_{max} + \lambda_{min})^2}{4 \lambda_{max} \lambda_{min}}, \quad \forall x \neq 0.$$

➤ This helps establish the convergence result

Let A an SPD matrix. Then, the A -norms of the error vectors $d_k = x_* - x_k$ generated by steepest descent satisfy:

$$\|d_{k+1}\|_A \leq \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}} \|d_k\|_A$$

➤ Algorithm converges for any initial guess x_0 .

Proof: Observe $\|d_{k+1}\|_A^2 = (Ad_{k+1}, d_{k+1}) = (r_{k+1}, d_{k+1})$

➤ by substitution,

$$\|d_{k+1}\|_A^2 = (r_{k+1}, d_k - \alpha_k r_k)$$

➤ By construction $r_{k+1} \perp r_k$ so we get $\|d_{k+1}\|_A^2 = (r_{k+1}, d_k)$. Now:

$$\begin{aligned}\|d_{k+1}\|_A^2 &= (r_k - \alpha_k Ar_k, d_k) \\ &= (r_k, A^{-1}r_k) - \alpha_k (r_k, r_k) \\ &= \|d_k\|_A^2 \left(1 - \frac{(r_k, r_k)}{(r_k, Ar_k)} \times \frac{(r_k, r_k)}{(r_k, A^{-1}r_k)} \right).\end{aligned}$$

Result follows by applying the Kantorovich inequality. ■


2. Minimal residual iteration.

A positive definite ($A + A^T$ is SPD). Take at each step $d = r$ and $e = Ar$.

Iteration:

$$\begin{aligned} r &\leftarrow b - Ax, \\ \alpha &\leftarrow (Ar, r) / (Ar, Ar) \\ x &\leftarrow x + \alpha r \end{aligned}$$

- Each step minimizes $f(x) = \|b - Ax\|_2^2$ in direction r .
- Converges under the condition that $A + A^T$ is SPD.

 As is formulated, the above algorithm would require 2 'matvecs' at each step. Reformulate it so that only one matvec is required

Convergence

Let A be a real positive definite matrix, and let

$$\mu = \lambda_{\min}(A + A^T)/2, \quad \sigma = \|A\|_2.$$

Then the residual vectors generated by the Min. Res. Algorithm satisfy:

$$\|r_{k+1}\|_2 \leq \left(1 - \frac{\mu^2}{\sigma^2}\right)^{1/2} \|r_k\|_2$$

➤ In this case Min. Res. converges for any initial guess x_0 .

Proof: Similar to steepest descent. Start with

$$\begin{aligned}\|r_{k+1}\|_2^2 &= (r_{k+1}, r_k - \alpha_k Ar_k) \\ &= (r_{k+1}, r_k) - \alpha_k (r_{k+1}, Ar_k).\end{aligned}$$

By construction, $r_{k+1} = r_k - \alpha_k Ar_k$ is $\perp Ar_k$, so:

$\|r_{k+1}\|_2^2 = (r_{k+1}, r_k) = (r_k - \alpha_k Ar_k, r_k)$. Then:

$$\begin{aligned}\|r_{k+1}\|_2^2 &= (r_k, r_k) - \alpha_k (Ar_k, r_k) \\ &= \|r_k\|_2^2 \left(1 - \frac{(Ar_k, r_k)}{(r_k, r_k)} \frac{(Ar_k, r_k)}{(Ar_k, Ar_k)} \right) \\ &= \|r_k\|_2^2 \left(1 - \frac{(Ar_k, r_k)^2}{(r_k, r_k)^2} \frac{\|r_k\|_2^2}{\|Ar_k\|_2^2} \right).\end{aligned}$$

Result follows from the inequalities $(Ax, x)/(x, x) \geq \mu > 0$ and $\|Ar_k\|_2 \leq \|A\|_2 \|r_k\|_2$. ■

3. Residual norm steepest descent.

A is arbitrary (nonsingular). Take at each step $d = A^T r$ and $e = Ad$.

Iteration:

$$\begin{aligned} r &\leftarrow b - Ax, d = A^T r \\ \alpha &\leftarrow \|d\|_2^2 / \|Ad\|_2^2 \\ x &\leftarrow x + \alpha d \end{aligned}$$

- Each step minimizes $f(x) = \|b - Ax\|_2^2$ in direction $-\nabla f$.
- Important Note: equivalent to usual steepest descent applied to normal equations $A^T Ax = A^T b$.
- Converges under the condition that A is nonsingular.

KRYLOV SUBSPACE METHODS

Motivation

- One-dimensional projection techniques:

$$x_{new} = x + \alpha d$$

where d = a certain direction.

- α is defined to optimize a certain function.
- Equivalently: determine α by an orthogonality constraint

In MR:

Example

$$x(\alpha) = x + \alpha d, \text{ with } d = b - Ax.$$

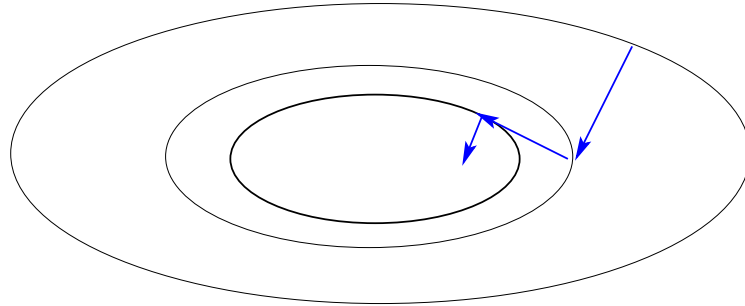
$$\min_{\alpha} \|b - Ax(\alpha)\|_2 \text{ reached iff } b - Ax(\alpha) \perp r$$

- One-dimensional projection methods are greedy methods. They are 'short-sighted'.

Example:

Recall in Steepest Descent: New direction of search \tilde{r} is \perp to old direction of search r .

$$\begin{aligned}r &\leftarrow b - Ax, \\ \alpha &\leftarrow (r, r) / (Ar, r) \\ x &\leftarrow x + \alpha r\end{aligned}$$



Question: can we do better by combining successive iterates?

➤ Yes: Krylov subspace methods..

Krylov subspace methods: Introduction

➤ Consider MR (or steepest descent). At each iteration:

$$\begin{aligned}r_{k+1} &= b - A(x^{(k)} + \alpha_k r_k) \\ &= r_k - \alpha_k A r_k \\ &= (I - \alpha_k A) r_k\end{aligned}$$

➤ In the end: $r_{k+1} = (I - \alpha_k A)(I - \alpha_{k-1} A) \cdots (I - \alpha_0 A) r_0 = p_{k+1}(A) r_0$ where $p_{k+1}(t)$ is a polynomial of degree $k + 1$ of the form

$$p_{k+1}(t) = 1 - tq_k(t)$$

 Show that: $x^{(k+1)} = x^{(0)} + q_k(A) r_0$, with $\deg(q_k) = k$

➤ Krylov subspace methods: iterations of this form that are ‘optimal’ [from m -dimensional projection methods]

Krylov subspace methods

Principle: Projection methods on Krylov subspaces:

$$K_m(A, v_1) = \text{span}\{v_1, Av_1, \dots, A^{m-1}v_1\}$$

- The most important class of iterative methods.
- Many variants exist depending on the subspace L .

Simple properties of K_m [$\mu \equiv \text{deg. of minimal polynomial of } v_1$.]

- $K_m = \{p(A)v_1 \mid p = \text{polynomial of degree } \leq m - 1\}$
- $K_m = K_\mu$ for all $m \geq \mu$. Moreover, K_μ is invariant under A .
- $\dim(K_m) = m$ iff $\mu \geq m$.

Arnoldi's algorithm

- Goal: to compute an orthogonal basis of K_m .
- Input: Initial vector v_1 , with $\|v_1\|_2 = 1$ and m .

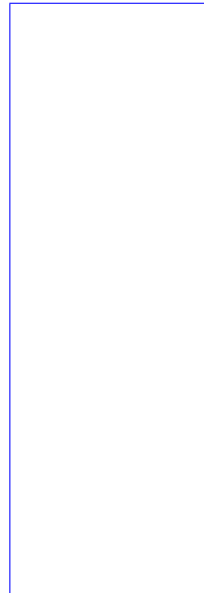
ALGORITHM : 1. *Arnoldi*

```
1: for  $j = 1, \dots, m$  do  
2:   Compute  $w := Av_j$   
3:   for  $i = 1, \dots, j$  do  
4:      $h_{i,j} := (w, v_i)$   
5:      $w := w - h_{i,j}v_i$   
6:   end for  
7:   Compute:  $h_{j+1,j} = \|w\|_2$  and  $v_{j+1} = w/h_{j+1,j}$   
8: end for
```

Result of orthogonalization process (Arnoldi):

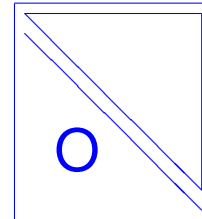
1. $V_m = [v_1, v_2, \dots, v_m]$ orthonormal basis of K_m .
2. $AV_m = V_{m+1}\overline{H}_m$
3. $V_m^T AV_m = H_m \equiv \overline{H}_m$ – last row.

$$V_m =$$



$$AV_m = V_{m+1}\overline{H}_m$$

$$\overline{H}_m =$$



$$V_{m+1} = [V_m, v_{m+1}]$$

Arnoldi's Method for linear systems ($L_m = K_m$)

From Petrov-Galerkin condition when $L_m = K_m$, we get

$$x_m = x_0 + V_m H_m^{-1} V_m^T r_0$$

➤ Select $v_1 = r_0 / \|r_0\|_2 \equiv r_0 / \beta$ in Arnoldi's. Then

$$x_m = x_0 + \beta V_m H_m^{-1} e_1$$

 What is the residual vector $r_m = b - Ax_m$?

Several algorithms mathematically equivalent to this approach:

* FOM [Y. Saad, 1981] (above formulation), Young and Jea's ORTHORES [1982], Axelsson's projection method [1981],..

* Also Conjugate Gradient method [see later]

Minimal residual methods ($L_m = AK_m$)

When $L_m = AK_m$, we let $W_m \equiv AV_m$ and obtain relation

$$\begin{aligned}x_m &= x_0 + V_m[W_m^T AV_m]^{-1}W_m^T r_0 \\ &= x_0 + V_m[(AV_m)^T AV_m]^{-1}(AV_m)^T r_0.\end{aligned}$$

► Use again $v_1 := r_0/(\beta := \|r_0\|_2)$ and the relation

$$AV_m = V_{m+1}\bar{H}_m$$

► $x_m = x_0 + V_m[\bar{H}_m^T \bar{H}_m]^{-1}\bar{H}_m^T \beta e_1 = x_0 + V_m y_m$
where y_m minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ over $y \in \mathbb{R}^m$.

- Gives the Generalized Minimal Residual method (GMRES) ([YS-Schultz,'86]):

$$\begin{aligned} \mathbf{x}_m &= \mathbf{x}_0 + \mathbf{V}_m \mathbf{y}_m \quad \text{where} \\ \mathbf{y}_m &= \min_y \|\beta \mathbf{e}_1 - \bar{\mathbf{H}}_m \mathbf{y}\|_2 \end{aligned}$$

- Several Mathematically equivalent methods:

- Axelsson's CGLS
- Orthomin (1980)
- Orthodir
- GCR

The symmetric case: Observation

Observe: When A is real symmetric then in Arnoldi's method:

$$H_m = V_m^T A V_m$$

must be symmetric. Therefore

Theorem. When Arnoldi's algorithm is applied to a (real) symmetric matrix then the matrix H_m is symmetric tridiagonal:

$$h_{ij} = 0 \quad 1 \leq i < j - 1; \quad \text{and}$$
$$h_{j,j+1} = h_{j+1,j}, \quad j = 1, \dots, m$$

➤ We can write

$$H_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \beta_m & \alpha_m & \end{bmatrix} \quad (1)$$

The v_i 's satisfy a 3-term recurrence [Lanczos Algorithm]:

$$\beta_{j+1}v_{j+1} = Av_j - \alpha_jv_j - \beta_jv_{j-1}$$

➤ Simplified version of Arnoldi's algorithm for sym. systems.

Symmetric matrix + Arnoldi → Symmetric Lanczos

The Lanczos algorithm

ALGORITHM : 2. Lanczos

1. Choose an initial vector v_1 , s.t. $\|v_1\|_2 = 1$
Set $\beta_1 \equiv 0, v_0 \equiv 0$
2. For $j = 1, 2, \dots, m$ Do:
3. $w_j := Av_j - \beta_j v_{j-1}$
4. $\alpha_j := (w_j, v_j)$
5. $w_j := w_j - \alpha_j v_j$
6. $\beta_{j+1} := \|w_j\|_2$. If $\beta_{j+1} = 0$ then Stop
7. $v_{j+1} := w_j / \beta_{j+1}$
8. EndDo

Lanczos algorithm for linear systems

➤ Usual orthogonal projection method setting:

- $L_m = K_m = \text{span}\{r_0, Ar_0, \dots, A^{m-1}r_0\}$
- Basis $V_m = [v_1, \dots, v_m]$ of K_m generated by the Lanczos algorithm

➤ Three different possible implementations.

(1) Arnoldi-like;

(2) Exploit tridiagonal nature of H_m (DIOM);

(3) Conjugate gradient (CG) - derived from (2)

We will skip details and show the CG algorithm

The Conjugate Gradient Algorithm (A S.P.D.)

ALGORITHM : 3. *Conjugate Gradient Method*

1: **Start:** $r_0 := b - Ax_0, p_0 := r_0.$

2: **while** (x_j Not-converged) **do**

3: $\alpha_j := (r_j, r_j) / (Ap_j, p_j)$

4: $x_{j+1} := x_j + \alpha_j p_j$

5: $r_{j+1} := r_j - \alpha_j Ap_j$

6: $\beta_j := (r_{j+1}, r_{j+1}) / (r_j, r_j)$

7: $p_{j+1} := r_{j+1} + \beta_j p_j$

8: **end while**

■ $r_j = scaling \times v_{j+1} \rightarrow$ the r_j 's are orthogonal.

■ The p_j 's are A -conjugate, i.e., $(Ap_i, p_j) = 0$ for $i \neq j$.

A bit of history. From the 1952 CG article:

“The method of conjugate gradients was developed independently by E. Stiefel of the Institute of Applied Mathematics at Zurich and by M. R. Hestenes with the cooperation of J. B. Rosser, G. Forsythe, and L. Paige of the Institute for Numerical Analysis, National Bureau of Standards. (...) The first papers on this method were given by E. Stiefel [1952] and by M. R. Hestenes [1951]. Reports on this method were given by E. Stiefel and J. B. Rosser at a Symposium on August 23-25, 1951. Recently, C. Lanczos [1952] developed a closely related routine based on his earlier paper on eigenvalue problem [1950]. Examples and numerical tests of the method have been by R. Hayes, U. Hoschstrasser, and M. Stein.”

SOLUTION OF EIGENVALUE PROBLEMS

Background. Origins of Eigenvalue Problems

- Structural Engineering [$Ku = \lambda Mu$] (Goal: frequency response)
 - Electronic structure calculations [Schrödinger equation..]
 - Stability analysis [e.g., electrical networks, mechanical system,..]
 - Bifurcation analysis [e.g., in fluid flow]
- Large eigenvalue problems in quantum chemistry use up biggest portion of the time in supercomputer centers

Background. New applications in data analytics

- Machine learning problems often require a (partial) *Singular Value Decomposition* -
- Somewhat different issues in this case:
 - Very large matrices, update the SVD
 - Compute dominant singular values/vectors
 - Many problems of approximating a matrix (or a **tensor**) by one of lower rank (Dimension reduction, ...)
- But: Methods for computing SVD often based on those for standard eigenvalue problems

Background. The Problem (s)

- Standard eigenvalue problem:

$$Ax = \lambda x$$

Often: A is symmetric real (or Hermitian complex)

- Generalized problem $Ax = \lambda Bx$ Often: B is symmetric positive definite, A is symmetric or nonsymmetric

- Quadratic problems:

$$(A + \lambda B + \lambda^2 C)u = 0$$

- Nonlinear eigenvalue problems (NEVP)

$$\left[A_0 + \lambda B_0 + \sum_{i=1}^n f_i(\lambda) A_i \right] u = 0$$

➤ General form of NEVP $A(\lambda)x = 0$

➤ Nonlinear **eigenvector** problems:

$$[A + \lambda B + F(u_1, u_2, \dots, u_k)]u = 0$$

What to compute:

- A few λ_i 's with smallest or largest real parts;
- All λ_i 's in a certain region of \mathbb{C} ;
- A few of the dominant eigenvalues;
- All λ_i 's (rare).

Large eigenvalue problems in applications

- Some applications require the computation of a large number of eigenvalues and vectors of very large matrices.
- Density Functional Theory in electronic structure calculations: '*ground states*'
- *Excited states* involve transitions and invariably lead to much more complex computations. → Large matrices, *many* eigen-pairs to compute

Background: The main tools

Projection process: Rayleigh-Ritz

- (a) Build a 'good' subspace $K = \text{span}(V)$;
- (b) get approximate eigenpairs by a Rayleigh-Ritz process:

Find $\tilde{\lambda} \in \mathbb{C}$, $\tilde{u} \in K$ such that: $(A - \tilde{\lambda}I)\tilde{u} \perp K$

➤ Will revisit this shortly

The main tools: Shift-and-invert:

- If we want eigenvalues near σ , replace A by $(A - \sigma I)^{-1}$.

Example: power method: $v_j = Av_{j-1}/\text{scaling}$ replaced by

$$v_j = \frac{(A - \sigma I)^{-1} v_{j-1}}{\text{scaling}}$$

- Works well for computing *a few* eigenvalues near σ /
- Used in commercial package NASTRAN (for decades!)
- Requires factoring $(A - \sigma I)$ (or $(A - \sigma B)$ in generalized case.) But convergence will be much faster.
- A solve each time - Factorization done once (ideally).

The main tools: Deflation / Restarting

Deflation: ➤ Once eigenvectors converge remove them from the picture (e.g., with power method, second largest becomes largest eigenvalue after deflation).

Restarting Strategies:

➤ Restart projection process by using information gathered in previous steps

➤ ALL available methods use some combination of these ingredients.

[e.g. ARPACK: Arnoldi/Lanczos + ‘implicit restarts’ + shift-and-invert (option).]

Current state-of-the art in eigensolvers

- Eigenvalues at one end of the spectrum:
 - Subspace iteration + filtering [e.g. **FEAST**, **Cheb**,...]
 - Lanczos+variants (no restart, thick restart, implicit restart, Davidson,..), e.g., **ARPACK** code, **PRIMME**.
 - Block Algorithms [Block Lanczos, **TraceMin**, **LOBPCG**, **SlepSc**,...]
 - + Many others - more or less related to above
- ‘Interior’ eigenvalue problems (middle of spectrum):
 - Combine shift-and-invert + Lanczos/block Lanczos. Used in, e.g., **NASTRAN**
 - Rational filtering [**FEAST**, Sakurai et al.,...]

Projection Methods for Eigenvalue Problems

General formulation:

- Projection method onto K orthogonal to L
- Given: Two subspaces K and L of same dimension.
- Find: $\tilde{\lambda}, \tilde{u}$ such that:

$$\tilde{\lambda} \in \mathbb{C}, \tilde{u} \in K; \quad (\tilde{\lambda}I - A)\tilde{u} \perp L$$

Two types of methods:

- Orthogonal projection methods: situation when $L = K$.
- Oblique projection methods: When $L \neq K$.

Rayleigh-Ritz projection

Given: a subspace X known to contain good approximations to eigenvectors of A .

Question: How to extract good approximations to eigenvalues/ eigenvectors from this subspace?

Answer: Rayleigh Ritz process.

Let $Q = [q_1, \dots, q_m]$ an orthonormal basis of X . Then write an approximation in the form $\tilde{u} = Qy$ and obtain y by writing

$$Q^H (A - \tilde{\lambda}I) \tilde{u} = 0$$

$$\triangleright Q^H A Q y = \tilde{\lambda} y$$

Procedure:

1. Obtain an orthonormal basis of X
2. Compute $C = Q^H A Q$ (an $m \times m$ matrix)
3. Obtain Schur factorization of C , $C = Y R Y^H$
4. Compute $\tilde{U} = Q Y$

Property: if X is (exactly) invariant, then procedure will yield exact eigenvalues and eigenvectors.

Proof: Since X is invariant, $(A - \tilde{\lambda}I)u = Qz$ for a certain z . $Q^H Qz = 0$ implies $z = 0$ and therefore $(A - \tilde{\lambda}I)u = 0$.

➤ Can use this procedure in conjunction with the subspace obtained from subspace iteration algorithm

Subspace Iteration

- *Original idea:* projection technique onto a subspace of the form $Y = A^k X$
- In practice: Replace A^k by suitable polynomial [Chebyshev]

Advantages:

- Easy to implement (in symmetric case);
- Easy to analyze;

Disadvantage: Slow.

- Often used with polynomial acceleration: $A^k X$ replaced by $C_k(A)X$. Typically $C_k =$ Chebyshev polynomial.

Algorithm: *Subspace Iteration with Projection*

1. **Start:** Choose an initial system of vectors $X = [x_0, \dots, x_m]$ and an initial polynomial C_k .

2. **Iterate:** Until convergence do:

(a) Compute $\hat{Z} = C_k(A)X_{old}$.

(b) Orthonormalize \hat{Z} into Z .

(c) Compute $B = Z^H A Z$ and use the QR algorithm to compute the Schur vectors $Y = [y_1, \dots, y_m]$ of B .

(d) Compute $X_{new} = ZY$.

(e) Test for convergence. If satisfied stop. Else select a new polynomial $C'_{k'}$ and continue.

THEOREM: Let $S_0 = \text{span}\{x_1, x_2, \dots, x_m\}$ and assume that S_0 is such that the vectors $\{Px_i\}_{i=1, \dots, m}$ are linearly independent where P is the spectral projector associated with $\lambda_1, \dots, \lambda_m$. Let \mathcal{P}_k the orthogonal projector onto the subspace $S_k = \text{span}\{X_k\}$. Then for each eigenvector u_i of A , $i = 1, \dots, m$, there exists a unique vector s_i in the subspace S_0 such that $Ps_i = u_i$. Moreover, the following inequality is satisfied

$$\|(I - \mathcal{P}_k)u_i\|_2 \leq \|u_i - s_i\|_2 \left(\left| \frac{\lambda_{m+1}}{\lambda_i} \right| + \epsilon_k \right)^k, \quad (2)$$

where ϵ_k tends to zero as k tends to infinity.

Krylov subspace methods

Principle: Projection methods on Krylov subspaces:

$$K_m(A, v_1) = \text{span}\{v_1, Av_1, \dots, A^{m-1}v_1\}$$

- The most important class of iterative methods.
- Many variants exist depending on the subspace L .

Simple properties of K_m [$\mu \equiv \text{deg. of minimal polynomial of } v_1$.]

- $K_m = \{p(A)v_1 \mid p = \text{polynomial of degree } \leq m - 1\}$
- $K_m = K_\mu$ for all $m \geq \mu$. Moreover, K_μ is invariant under A .
- $\dim(K_m) = m$ iff $\mu \geq m$.

Arnoldi's Algorithm

- Goal: to compute an orthogonal basis of K_m .
- Input: Initial vector v_1 , with $\|v_1\|_2 = 1$ and m .

ALGORITHM : 4. *Arnoldi's procedure*

For $j = 1, \dots, m$ *do*

Compute $w := Av_j$

For $i = 1, \dots, j$, *do* $\left\{ \begin{array}{l} h_{i,j} := (w, v_i) \\ w := w - h_{i,j}v_i \end{array} \right.$

$h_{j+1,j} = \|w\|_2; v_{j+1} = w/h_{j+1,j}$

End

Result of Arnoldi's algorithm

Let

$$\overline{H}_m = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{bmatrix}; \quad H_m = \overline{H}_m(1:m, 1:m)$$

1. $V_m = [v_1, v_2, \dots, v_m]$ orthonormal basis of K_m .
2. $AV_m = V_{m+1}\overline{H}_m = V_m H_m + h_{m+1,m}v_{m+1}e_m^T$
3. $V_m^T AV_m = H_m \equiv \overline{H}_m - \text{last row.}$

Application to eigenvalue problems

- Write approximate eigenvector as $\tilde{u} = V_m y$ + Galerkin condition

$$(A - \tilde{\lambda}I)V_m y \perp \mathcal{K}_m \rightarrow V_m^H (A - \tilde{\lambda}I)V_m y = 0$$

- Approximate eigenvalues are eigenvalues of H_m

$$H_m y_j = \tilde{\lambda}_j y_j$$

Associated approximate eigenvectors are

$$\tilde{u}_j = V_m y_j$$

Typically a few of the outermost eigenvalues will converge first.

Restarted Arnoldi

In practice: Memory requirement of algorithm implies restarting is necessary

➤ Restarted Arnoldi for computing rightmost eigenpair:

ALGORITHM : 5 . Restarted Arnoldi

1. **Start:** Choose an initial vector v_1 and a dimension m .
2. **Iterate:** Perform m steps of Arnoldi's algorithm.
3. **Restart:** Compute the approximate eigenvector $u_1^{(m)}$
4. associated with the rightmost eigenvalue $\lambda_1^{(m)}$.
5. If satisfied stop, else set $v_1 \equiv u_1^{(m)}$ and goto 2.

Deflation

- Very useful in practice.
- Different forms: locking (subspace iteration), selective orthogonalization (Lanczos), Schur deflation, ...

A little background

Consider Schur canonical form $A = URU^H$

where U is a (complex) upper triangular matrix.

- Vector columns u_1, \dots, u_n called **Schur vectors**.
- Note: Schur vectors are not unique. In particular, they depend on the order of the eigenvalues

Wiedlandt Deflation: Assume we have computed a right eigenpair λ_1, u_1 .
Wielandt deflation considers eigenvalues of

$$A_1 = A - \sigma u_1 v^H$$

Note:

$$\Lambda(A_1) = \{\lambda_1 - \sigma, \lambda_2, \dots, \lambda_n\}$$

Wielandt deflation preserves u_1 as an eigenvector as well all the left eigenvectors not associated with λ_1 .

- An interesting choice for v is to take simply $v = u_1$. In this case Wielandt deflation preserves Schur vectors as well.
- Can apply above procedure successively.

ALGORITHM : 6. *Explicit Deflation*

1. $A_0 = A$
2. For $j = 0 \dots \mu - 1$ Do:
3. Compute a dominant eigenvector of A_j
4. Define $A_{j+1} = A_j - \sigma_j u_j u_j^H$
5. End

- Computed u_1, u_2, \dots form a set of Schur vectors for A .
- In Arnoldi: Accumulate each new converged eigenvector in columns 1, 2, 3, ... ['locked' set of eigenvectors.] + maintain orthogonality
- Alternative: implicit deflation (within a procedure such as Arnoldi).

Deflated Arnoldi

For $k = 1, \dots, NEV$ do: /* Eigenvalue loop */

1. For $j = k, k + 1, \dots, m$ do: /* Arnoldi loop*/

- Compute $w := Av_j$.
- Orthonormalize w against $v_1, v_2, \dots, v_j \rightarrow v_{j+1}$

2. Compute next approximate eigenpair $\tilde{\lambda}, \tilde{u}$.

3. Orthonormalize \tilde{u} against v_1, \dots, v_j ➤ Result = \tilde{s} = approximate Schur vector.

4. Define $v_k := \tilde{s}$.

5. If approximation not satisfactory go to 1.

6. Else define $h_{i,k} = (Av_k, v_i)$, $i = 1, \dots, k$,

Thus, for $k = 2$:

$$V_m = \left[\underbrace{v_1, v_2}_{\text{Locked}}, \overbrace{v_3, \dots, v_m}^{\text{active}} \right]$$

$$H_m = \left(\begin{array}{cc|cccc} * & * & * & * & * & * \\ & * & * & * & * & * \\ \hline & & * & * & * & * \\ & & * & * & * & * \\ & & & * & * & * \\ & & & & * & * \end{array} \right)$$

➤ Similar techniques in Subspace iteration [G. Stewart's SRRIT]

Hermitian case: The Lanczos Algorithm

- The Hessenberg matrix becomes tridiagonal :

$$A = A^H \quad \text{and} \quad V_m^H A V_m = H_m \quad \rightarrow \quad H_m = H_m^H \quad \longrightarrow$$

$$H_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \beta_m & \alpha_m & \\ & & & & & & \end{bmatrix}$$

Consequence:

3-term recurrence:

$$\beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}$$

Hermitian matrix + Arnoldi \rightarrow Hermitian Lanczos

ALGORITHM : 7. *Lanczos*

1. Choose v_1 of norm unity. Set $\beta_1 \equiv 0, v_0 \equiv 0$
2. For $j = 1, 2, \dots, m$ Do:
3. $w_j := Av_j - \beta_j v_{j-1}$
4. $\alpha_j := (w_j, v_j)$
5. $w_j := w_j - \alpha_j v_j$
6. $\beta_{j+1} := \|w_j\|_2$. If $\beta_{j+1} = 0$ then Stop
7. $v_{j+1} := w_j / \beta_{j+1}$
8. EndDo

- In theory v_i 's defined by 3-term recurrence are orthogonal.
- However: in practice severe loss of orthogonality;

Lanczos with reorthogonalization

Observation [Paige, 1981]: Loss of orthogonality starts suddenly, when the first eigenpair converges. It indicates loss of linear independence of the v_i s. When orthogonality is lost, then several copies of the same eigenvalue start appearing.

Forms of Re-orthogonalization

Full – reorthogonalize v_{j+1} against all previous v_i 's every time.

Partial – reorthogonalize v_{j+1} against all previous v_i 's only when needed

Selective – reorthogonalize v_{j+1} against computed eigenvectors

None – Do not reorthogonalize - but take measures to deal with 'spurious' eigenvalues.

The Lanczos Algorithm in the Hermitian Case

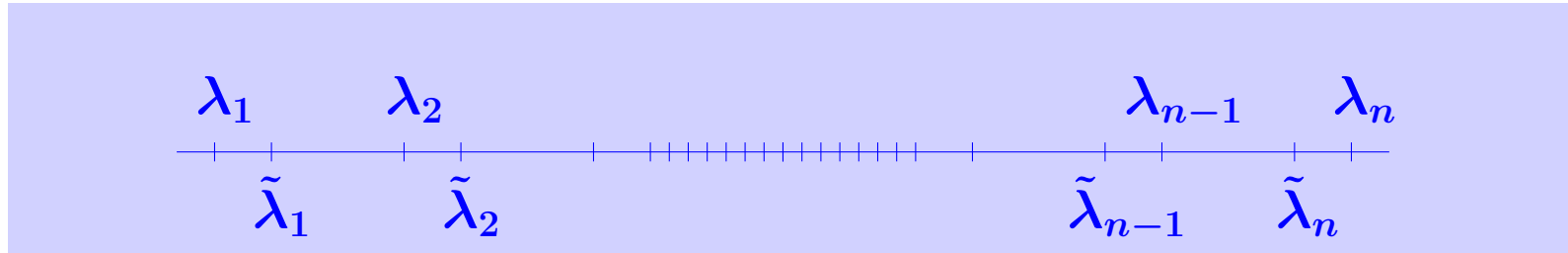
Assume eigenvalues sorted increasingly

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$

- Orthogonal projection method onto K_m ;
- To derive error bounds, use the Courant characterization

$$\tilde{\lambda}_1 = \min_{u \in K, u \neq 0} \frac{(Au, u)}{(u, u)} = \frac{(A\tilde{u}_1, \tilde{u}_1)}{(\tilde{u}_1, \tilde{u}_1)}$$
$$\tilde{\lambda}_j = \min_{\substack{u \in K, u \neq 0 \\ u \perp \tilde{u}_1, \dots, \tilde{u}_{j-1}}} \frac{(Au, u)}{(u, u)} = \frac{(A\tilde{u}_j, \tilde{u}_j)}{(\tilde{u}_j, \tilde{u}_j)}$$

- Bounds for λ_1 easy to find – similar to linear systems.
- Ritz values approximate eigenvalues of A inside out:



A-priori error bounds

Theorem [Kaniel, 1966]: Let $\gamma_1 = \frac{\lambda_2 - \lambda_1}{\lambda_N - \lambda_2}$; Then:

$$0 \leq \lambda_1^{(m)} - \lambda_1 \leq (\lambda_N - \lambda_1) \left[\frac{\tan \angle(v_1, u_1)}{T_{m-1}(1 + 2\gamma_1)} \right]^2$$

Theorem [Kaniel, Paige, YS]. Let $\gamma_i = \frac{\lambda_{i+1} - \lambda_i}{\lambda_N - \lambda_{i+1}}$, $\kappa_i^{(m)} = \prod_{j < i} \frac{\lambda_j^{(m)} - \lambda_N}{\lambda_j^{(m)} - \lambda_i}$ Then:

$$0 \leq \lambda_i^{(m)} - \lambda_i \leq (\lambda_N - \lambda_i) \left[\kappa_i^{(m)} \frac{\tan \angle(v_i, u_i)}{T_{m-i}(1 + 2\gamma_i)} \right]^2$$

The Lanczos biorthogonalization ($A^H \neq A$)

ALGORITHM : 8. Lanczos bi-orthogonalization

1. Choose two vectors v_1, w_1 such that $(v_1, w_1) = 1$.
2. Set $\beta_1 = \delta_1 \equiv 0, w_0 = v_0 \equiv 0$
3. For $j = 1, 2, \dots, m$ Do:
4. $\alpha_j = (Av_j, w_j)$
5. $\hat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
6. $\hat{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$
7. $\delta_{j+1} = |(\hat{v}_{j+1}, \hat{w}_{j+1})|^{1/2}$. If $\delta_{j+1} = 0$ Stop
8. $\beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1}) / \delta_{j+1}$
9. $w_{j+1} = \hat{w}_{j+1} / \beta_{j+1}$
10. $v_{j+1} = \hat{v}_{j+1} / \delta_{j+1}$
11. EndDo

- Builds a pair of biorthogonal bases for the two subspaces

$$\mathcal{K}_m(A, v_1) \quad \text{and} \quad \mathcal{K}_m(A^H, w_1)$$

- Many choices for $\delta_{j+1}, \beta_{j+1}$ in lines 7 and 8. Only constraint:

$$\delta_{j+1}\beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1})$$

Let

$$T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \delta_2 & \alpha_2 & \beta_3 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m & \\ & & & \delta_m & \alpha_m & \end{bmatrix} \cdot$$

- $v_i \in \mathcal{K}_m(A, v_1)$ and $w_j \in \mathcal{K}_m(A^T, w_1)$.

If the algorithm does not break down before step m , then the vectors $v_i, i = 1, \dots, m$, and $w_j, j = 1, \dots, m$, are biorthogonal, i.e.,

$$(v_j, w_i) = \delta_{ij} \quad 1 \leq i, j \leq m .$$

Moreover, $\{v_i\}_{i=1,2,\dots,m}$ is a basis of $\mathcal{K}_m(A, v_1)$ and $\{w_i\}_{i=1,2,\dots,m}$ is a basis of $\mathcal{K}_m(A^H, w_1)$ and

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^H,$$

$$A^H W_m = W_m T_m^H + \bar{\beta}_{m+1} w_{m+1} e_m^H,$$

$$W_m^H AV_m = T_m .$$

➤ If θ_j, y_j, z_j are, respectively an eigenvalue of T_m , with associated right and left eigenvectors y_j and z_j respectively, then corresponding approximations for A are

Ritz value	Right Ritz vector	Left Ritz vector
θ_j	$V_m y_j$	$W_m z_j$

[Note: terminology is abused slightly - Ritz values and vectors normally refer to Hermitian cases.]

Advantages and disadvantages

Advantages:

- Nice three-term recurrence – requires little storage in theory.
- Computes left and a right eigenvectors at the same time

Disadvantages:

- Algorithm can break down or nearly break down.
- Convergence not too well understood. Erratic behavior
- Not easy to take advantage of the tridiagonal form of T_m .

Look-ahead Lanczos

Algorithm breaks down when:

$$(\hat{v}_{j+1}, \hat{w}_{j+1}) = 0$$

Three distinct situations.

- ‘lucky breakdown’ when either \hat{v}_{j+1} or \hat{w}_{j+1} is zero. In this case, eigenvalues of T_m are eigenvalues of A .
- $(\hat{v}_{j+1}, \hat{w}_{j+1}) = 0$ but of $\hat{v}_{j+1} \neq 0$, $\hat{w}_{j+1} \neq 0 \rightarrow$ **serious breakdown**. Often possible to bypass the step (+ a few more) and continue the algorithm. If this is not possible then we get an ...
- ... Incurable break-down. [very rare]