Rational Krylov methods for linear and nonlinear eigenvalue problems

Mele Giampaolo
mele@mail.dm.unipi.it

University of Pisa

7 March 2014

Outline

- Arnoldi (and its variants) for linear eigenproblems
- Rational Krylov algorithm for linear eigenproblems
- Applications of Rational Krylov algorithm for nonlinear eigenproblems
  - Linearization by means of Hermite interpolation
  - Iterative projection methods

Outline

- Arnoldi (and its variants) for linear eigenproblems
- Rational Krylov algorithm for linear eigenproblems
- Applications of Rational Krylov algorithm for nonlinear eigenproblems
  - Linearization by means of Hermite interpolations
  - Iterative projection methods

Eigenvalue problem

Definition of the problem

Given an application

$$A(\cdot) : \mathbb{C} \to \mathbb{C}^{n \times n}$$

find a pair $(\lambda, x) \in \mathbb{C} \times \mathbb{C}^n$ such that

$$A(\lambda)x = 0$$

The number λ is called eigenvalue
The vector x is called eigenvector

In practical applications the task is to compute eigenvalues in a subset $\Omega \subset \mathbb{C}$.

Linear eigenvalue problem

If $A(\lambda)$ is linear we call the problem *linear eigenvalue problem*

Let

$$A(\lambda) = A - \lambda B \qquad A, B \in \mathbb{C}^{n \times n}$$

then the linear eigenvalue problem is

$$Ax = \lambda Bx \qquad \text{Generalized eigenvalue problem}$$

in case $B = I$ (identity matrix)

$$Ax = \lambda x \qquad \text{Classic eigenvalue problem}$$

Arnoldi's algorithm for classic eigenvalue problem

Classic eigenvalue problem

Given $A \in \mathbb{C}^{n \times n}$ the problem is to compute the pairs (λ, x) such that $Ax = \lambda x$

Definition (Krylov subspace)

Given a vector $x \in \mathbb{C}^n$ and a natural number m

$$K_m(A, x) = \text{span}\left(x, Ax, A^2x, \ldots, A^{m-1}x\right)$$

is the Krylov subspace

The idea is to project the matrix A in the Krylov subspace and solve the projected problem.

Arnoldi's algorithm for classic eigenvalue problem

Gram–Schmidt orthogonalization

Given $x \in \mathbb{C}^n$ define

$$\begin{cases} v_1 := x/\|x\| & \\ h_{i,j} = Av_j \cdot v_i & i = 1, \dots j \\ w_{j+1} := Av_j - h_{1,j}v_1 - h_{2,j}v_2 - \cdots - h_{j,j}v_j & \\ h_{j+1,j} = \|w_{j+1}\| & \\ v_{j+1} = w_{j+1}/h_{j+1,j} & \end{cases}$$

Then $v_1, \dots, v_m$ is an orthonormal basis of $K_m(A, x)$.

Arnoldi sequence

In a vectorial form

$$AV_m = V_{m+1}H_{m+1,m}$$

Arnoldi's algorithm for classic eigenvalue problem

Observation

The matrix $H_{m,m}$ is the projection of A in $K_m(A, x)$, that is

$$V_m^H A V_m = H_{m,m}$$

Definition

Given an eigenpair (θ, s) of $H_{m,m}$, the value θ is called Ritz value and the vector $V_m s$ Ritz vector.

Proposition

If (θ, s) is an eigenpair of $H_{m,m}$ then

$$AV_m s - \theta V_m s = h_{m+1,m} s_m v_{m+1}.$$

If $h_{m+1,m} y_m$ is small, then $(\theta, V_m s)$ is an approximation of an eigenpair of A.

Arnoldi's algorithm for classic eigenvalue problem

Arnoldi's algorithm

1: Chose a starting vector x
2: **for** $m = 1, \ldots,$ till convergence **do**
3: Compute the Arnoldi sequence $AV_m = V_{m+1}H_{m+1,m}$
4: Compute eigenpairs (θ_i, y_i) of $H_{m,m}$
5: **if** $|h_{m+1,m}(e_m^H y_i)| < tol$ **then**
6: Store $(\theta_i, V_m y_i)$ as approximation of an eigenpair of A
7: **end if**
8: **end for**

Questions:

- How big must be m to get a good approximation of an eigenpair?
- How to choose a starting vector x?
- Which eigenpairs will be firstly approximated?

Arnoldi's algorithm for classic eigenvalue problem

Arnoldi's algorithm

1: Chose a starting vector x
2: **for** $m = 1, \ldots,$ till convergence **do**
3: Compute the Arnoldi sequence $AV_m = V_{m+1}H_{m+1,m}$
4: Compute eigenpairs (θ_i, y_i) of $H_{m,m}$
5: **if** $|h_{m+1,m}(e_m^H y_i)| < tol$ **then**
6: Store $(\theta_i, V_m y_i)$ as approximation of an eigenpair of A
7: **end if**
8: **end for**

Questions:

- How big must be m to get a good approximation of an eigenpair?
- How to choose a starting vector x?
- Which eigenpairs will be firstly approximated?

Convergence of Arnoldi's algorithm

Theorem (Saad)

If A is diagonalizable and (λ_i, u_i) are the eigenpairs, if

$$|\lambda_k - \lambda_1| > |\lambda_k - \lambda_j| \qquad \forall k \neq 1, j \neq 1$$

then λ_1 is the first eigenvalue to be approximated. Moreover the closer x to the eigenvector u_1 the faster the convergence to u_1.

In other words (under the hypothesis of the theorem) the outermost eigenvalues will be firstly approximated.

Heuristically, after a few steps, the approximations to the eigenvalues start to convergence linearly.

Convergence of Arnoldi's algorithm

Theorem (Saad)

If A is diagonalizable and (λ_i, u_i) are the eigenpairs, if

$$|\lambda_k - \lambda_1| > |\lambda_k - \lambda_j| \qquad \forall k \neq 1, j \neq 1$$

then λ_1 is the first eigenvalue to be approximated. Moreover the closer x to the eigenvector u_1 the faster the convergence to u_1.

In other words (under the hypothesis of the theorem) the outermost eigenvalues will be firstly approximated.

Heuristically, after a few steps, the approximations to the eigenvalues start to convergence linearly.

Thick restart

Problem

When the Arnoldi sequence grows too long, every step of the Arnoldi iteration gets slower. Moreover orthogonality is numerically lost.

Thick restart

Let $AV_m = V_{m+1}H_{m+1,m}$ be an Arnoldi sequence with $\theta_1, \ldots, \theta_k$ a subset of Ritz values, where at least one has not (numerically) converged yet. Then it is possible to build another Arnoldi sequence $AW_k = W_{k+1}\widetilde{H}_{k+1,k}$ such that $\theta_1, \ldots, \theta_k$ are the Ritz values.

The generation of the new sequence is numerically stable since it is done using Householder transformations.

The idea of thick restart is to select the Ritz values which we want to refine and remove the others.

- Lock
- Purge

Thick restart

Problem

When the Arnoldi sequence grows too long, every step of the Arnoldi iteration gets slower. Moreover orthogonality is numerically lost.

Thick restart

Let $AV_m = V_{m+1}H_{m+1,m}$ be an Arnoldi sequence with $\theta_1, \ldots, \theta_k$ a subset of Ritz values, where at least one has not (numerically) converged yet. Then it is possible to build another Arnoldi sequence $AW_k = W_{k+1}\widetilde{H}_{k+1,k}$ such that $\theta_1, \ldots, \theta_k$ are the Ritz values.

The generation of the new sequence is numerically stable since it is done using Householder transformations.

The idea of thick restart is to select the Ritz values which we want to refine and remove the others.

- Lock
- Purge

Arnoldi's algorithm for the linear eigenvalue problem

Tthe linear eigenproblem

$$Ax = \lambda Bx$$

can be solved by using Arnoldi's algorithm applied to the matrix $B^{-1}A$

- Matrices are often sparse/structured.
- B^{-1} is never computed.
- At each step of the algorithm a linear systems with the matrix B must be solved.
- The LU factorization of B can be performed once for all.

Shifted–and–inverted Arnoldi's algorithm for the linear eigenvalue problem

Proposition

If (θ, x) is an eigenpair of $(A - \sigma B)^{-1}B$ then $(\sigma + 1/\theta, x)$ is an eigenpair of the linear problem

$$Ax = \lambda Bx.$$

Observation

If θ is one of the outermost eigenvalues of $(A - \sigma B)^{-1}B$ then $\sigma + 1/\theta$ is one of the eigenvalues of the linear problem nearest σ. [Saad theorem].

This strategy can be used to compute eigenvalues of the linear problem near a point σ. If we want to compute eigenvalues in $\Omega \subset \mathbb{C}$ then we can select a few (equidistributed) points $\sigma_0, \ldots, \sigma_t \in \Omega$ and use this approach.

Shifted–and–inverted Arnoldi's algorithm for the linear eigenvalue problem

Proposition

If (θ, x) is an eigenpair of $(A - \sigma B)^{-1}B$ then $(\sigma + 1/\theta, x)$ is an eigenpair of the linear problem

$$Ax = \lambda Bx.$$

Observation

If θ is one of the outermost eigenvalues of $(A - \sigma B)^{-1}B$ then $\sigma + 1/\theta$ is one of the eigenvalues of the linear problem nearest σ. [Saad theorem].

This strategy can be used to compute eigenvalues of the linear problem near a point σ. If we want to compute eigenvalues in $\Omega \subset \mathbb{C}$ then we can select a few (equidistributed) points $\sigma_0, \ldots, \sigma_t \in \Omega$ and use this approach.

Outline

- Arnoldi (and its variants) for linear eigenproblems
- Rational Krylov algorithm for linear eigenproblems
- Applications of Rational Krylov algorithm for nonlinear eigenproblems
  - Linearization by means of Hermite interpolation
  - Iterative projection methods

Rational Krylov algorithm for linear eigenvalue problem

Problem

To compute eigenvalues near a set of points $\sigma_0, \ldots, \sigma_t \in \Omega$, one needs to apply Shifted–and–inverted Arnoldi's algorithm to each σ_i

Theorem (Ruhe)

In $O(m^3)$ it is possible change shift in the Arnoldi sequence, in particular

$$(A - \sigma_0 B)^{-1} B V_m = V_{m+1} H_{m+1,m} \Longrightarrow (A - \sigma_1 B)^{-1} B W_m = W_{m+1} \widetilde{H}_{m+1,m}$$

moreover $\operatorname{span}(V_{m+1}) = \operatorname{span}(W_{m+1})$. These operations are numerically stable if σ_0 and σ_1 are far enough from the eigenvalues of the original problem.

Rational Krylov algorithm for linear eigenvalue problem

Problem

To compute eigenvalues near a set of points $\sigma_0, \ldots, \sigma_t \in \Omega$, one needs to apply Shifted–and–inverted Arnoldi's algorithm to each σ_i

Theorem (Ruhe)

In $O(m^3)$ it is possible change shift in the Arnoldi sequence, in particular

$$(A - \sigma_0 B)^{-1} B V_m = V_{m+1} H_{m+1,m} \Longrightarrow (A - \sigma_1 B)^{-1} B W_m = W_{m+1} \widetilde{H}_{m+1,m}$$

moreover $\operatorname{span}(V_{m+1}) = \operatorname{span}(W_{m+1})$. These operations are numerically stable if σ_0 and σ_1 are far enough from the eigenvalues of the original problem.

Rational Krylov algorithm for linear eigenvalue problem

Rational Krylov algorithm

1: Chose a starting vector x and a starting shift σ_0 and define $v_1 = x/\|x\|$.
2: **for** $i = 1, \ldots,$ till convergence **do**
3: Extend the Arnoldi sequence $(A - \sigma_i B)^{-1} B V_m = V_{m+1} H_{m+1,m}$ till enough Ritz values near σ_i numerically converge. When needed, perform a thick restart.
4: Chose the next shift σ_{i+1} and transform the previous Arnoldi sequence in $(A - \sigma_{i+1} B)^{-1} B V_m = V_{m+1} H_{m+1,m}$ ny using $O(m^3)$ ops.
5: **end for**

Practical issues

- When shift changes, an LU factorization of $(A - \sigma_{i+1} B)$ is performed
- Heuristically, a good choice of the next shift is taking the average of *cstep* (small) Ritz values not yet converged and near the previous shift.
- Thick restart is performed.

Rational Krylov algorithm for linear eigenvalue problem

Rational Krylov algorithm

1: Chose a starting vector x and a starting shift σ_0 and define $v_1 = x/\|x\|$.
2: **for** $i = 1, \ldots,$ till convergence **do**
3: Extend the Arnoldi sequence $(A - \sigma_i B)^{-1} B V_m = V_{m+1} H_{m+1,m}$ till enough Ritz values near σ_i numerically converge. When needed, perform a thick restart.
4: Chose the next shift σ_{i+1} and transform the previous Arnoldi sequence in $(A - \sigma_{i+1} B)^{-1} B V_m = V_{m+1} H_{m+1,m}$ ny using $O(m^3)$ ops.
5: **end for**

Practical issues

- When shift changes, an LU factorization of $(A - \sigma_{i+1} B)$ is performed
- Heuristically, a good choice of the next shift is taking the average of *cstep* (small) Ritz values not yet converged and near the previous shift.
- Thick restart is performed.

Numerical experimentation

Tubolar reactor model

The conservation of reactant and energy in a homogeneous tube of length L in dimensionless form is modeled by

$$\begin{cases} \dfrac{L}{v}\dfrac{dy}{dt} &= -\dfrac{1}{Pe_m}\dfrac{\partial^2 y}{\partial X^2} + \dfrac{\partial y}{\partial X} + Dye^{\gamma-\gamma T^{-1}}, \\ \dfrac{L}{v}\dfrac{dT}{dt} &= -\dfrac{1}{Pe_h}\dfrac{\partial^2 T}{\partial X^2} + \dfrac{\partial T}{\partial X} + \beta(T-T_0) - BDye^{\gamma-\gamma T^{-1}}, \end{cases}$$

$$\text{B.C. :} \qquad y'(0) = Pe_m y(0),\ T'(0) = Pe_h T(0),\ y'(1) = 0,\ T'(1) = 0.$$

Where y is the concentration, T the temperature and $0 \leq X \leq 1$ the spatial coordinate. The setting of the problem is
$Pe_m = Pe_h = 5, B = 0.5, \gamma = 25, \beta = 3,5, D = 0,2662$ and $L/v = 1$.

The task is to solve numerically the equation with the method of lines.

Numerical experimentation

Stability of the time discretization

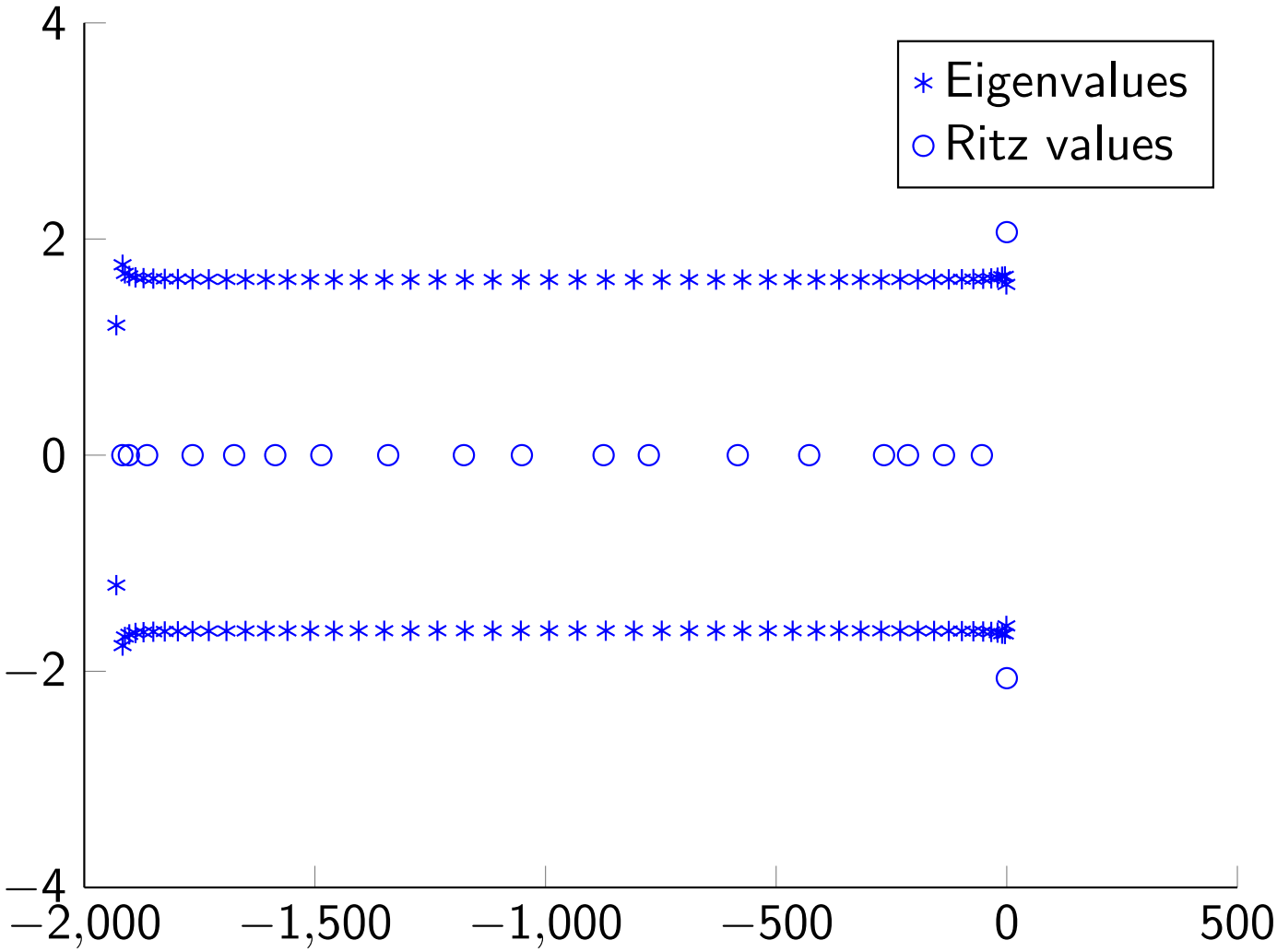
With a semi-discretization in space, setting $x = (y_1, T1, y_2, T_2, \ldots, y_{N/2}, T_{N/2})$ we get

$$\frac{d}{dt}\mathbf{x} = Ax \qquad A \in \mathbb{R}^{2N \times 2N},$$

where $h = 1/N$ is the discretization step. A is a banded matrix with bandwidth 5. In order to chose a stable time discretization it is needed to compute the rightmost eigenvalues of A.

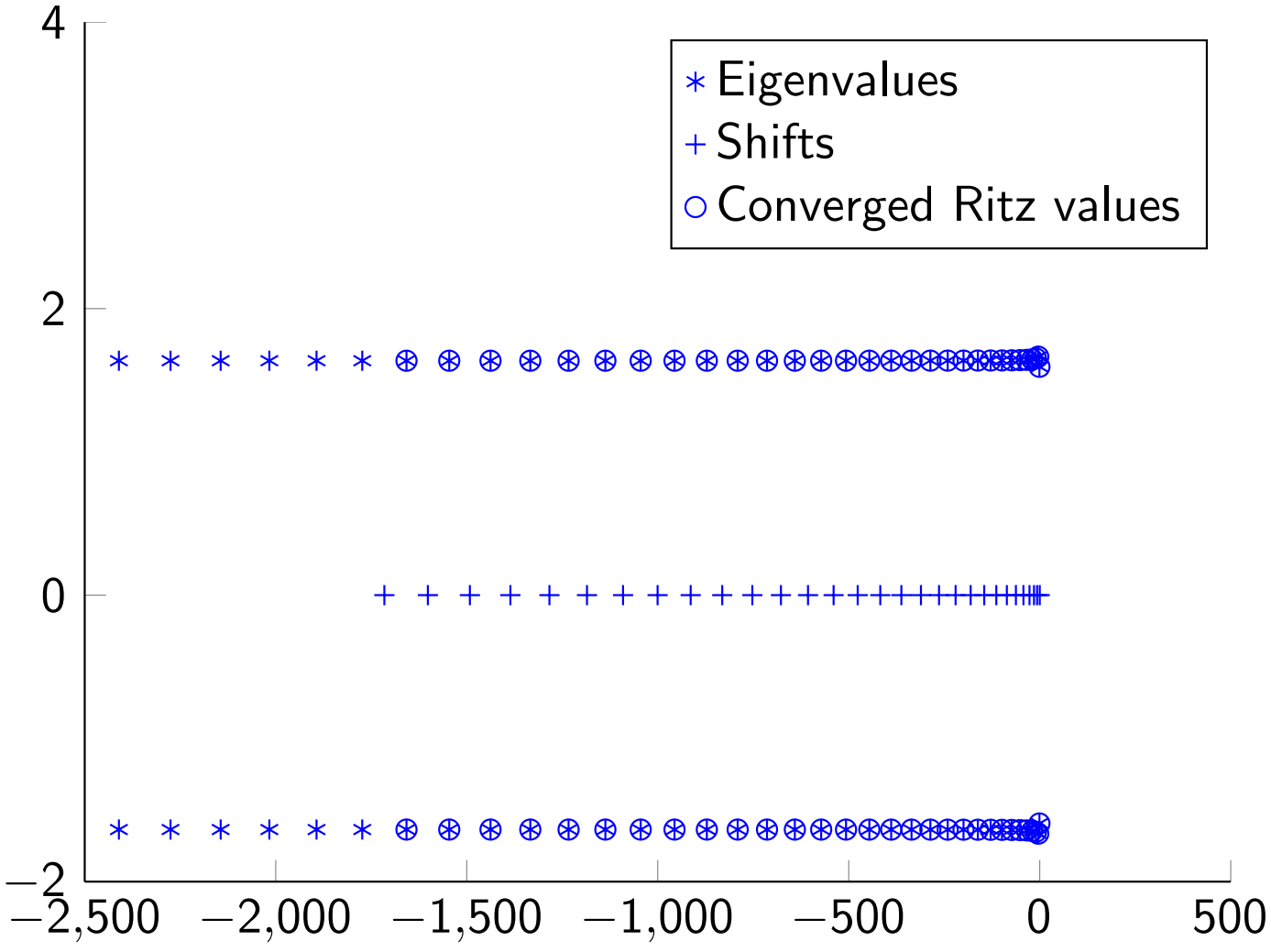
Numerical experimentation

$N = 50$, Arnoldi with 20 iterations.



Numerical experimentation

$N = 500$, Rational Krylov algorithm to compute 60 rightmost eigenvalues



Numerical experimentation

Convergence of the rightmost eigenvalues with Shift-and-inverted Arnoldi and with Rational Krylov

Wanted eigenvalues	Shift–and–inverted (number of steps)	Rational Krylov (number of steps)	Savings percentage (steps)
20	45	38	16 %
40	79	64	19 %
60	112	89	21 %
80	144	113	22 %

The advantage seems light, but with Rational Krylov method we can perform a thick restart. With shifted–and–inverted Arnoldi the restart induces a loop.

Numerical experimentation

Stability of a flow in a pipe

$$\begin{cases} \left\{(D^2-\alpha)^2 - i\alpha Re[\mathcal{U}_0(D^2-\alpha^2) - \mathcal{U}_0'']\right\}\tilde{v} = -ic\alpha Re(D^2-\alpha^2)\tilde{v} \\ \tilde{v}(1) = 0, \qquad D\tilde{v}(1)y = 0 \\ \tilde{v}(-1) = 0, \qquad D\tilde{v}(-1) = 0 \end{cases}$$

The setting is $\alpha = 1$ and $Re = 10000$.

Discrete problem

Using finite differences, we discretized with discretization step $h = 1/N$

$$A\underline{\tilde{v}} = cB\underline{\tilde{v}}$$

Where $A, B \in \mathbb{R}^{N\times N}$, $\det(A) \neq 0$, $\operatorname{rank}(B) = N - 4$ because of B.C.
A and B are banded matrices with bandwidth respectively 5 and 3.

The spectrum of the continuum problem has a branch structure, in particular it looks like a Y. The task is to compute the branch connected to zero.

Numerical experimentation

Stability of a flow in a pipe

$$\begin{cases} \left\{(D^2-\alpha)^2 - i\alpha Re[\mathcal{U}_0(D^2-\alpha^2) - \mathcal{U}_0'']\right\}\tilde{v} = -ic\alpha Re(D^2-\alpha^2)\tilde{v} \\ \tilde{v}(1) = 0, \qquad D\tilde{v}(1)y = 0 \\ \tilde{v}(-1) = 0, \qquad D\tilde{v}(-1) = 0 \end{cases}$$

The setting is $\alpha = 1$ and $Re = 10000$.

Discrete problem

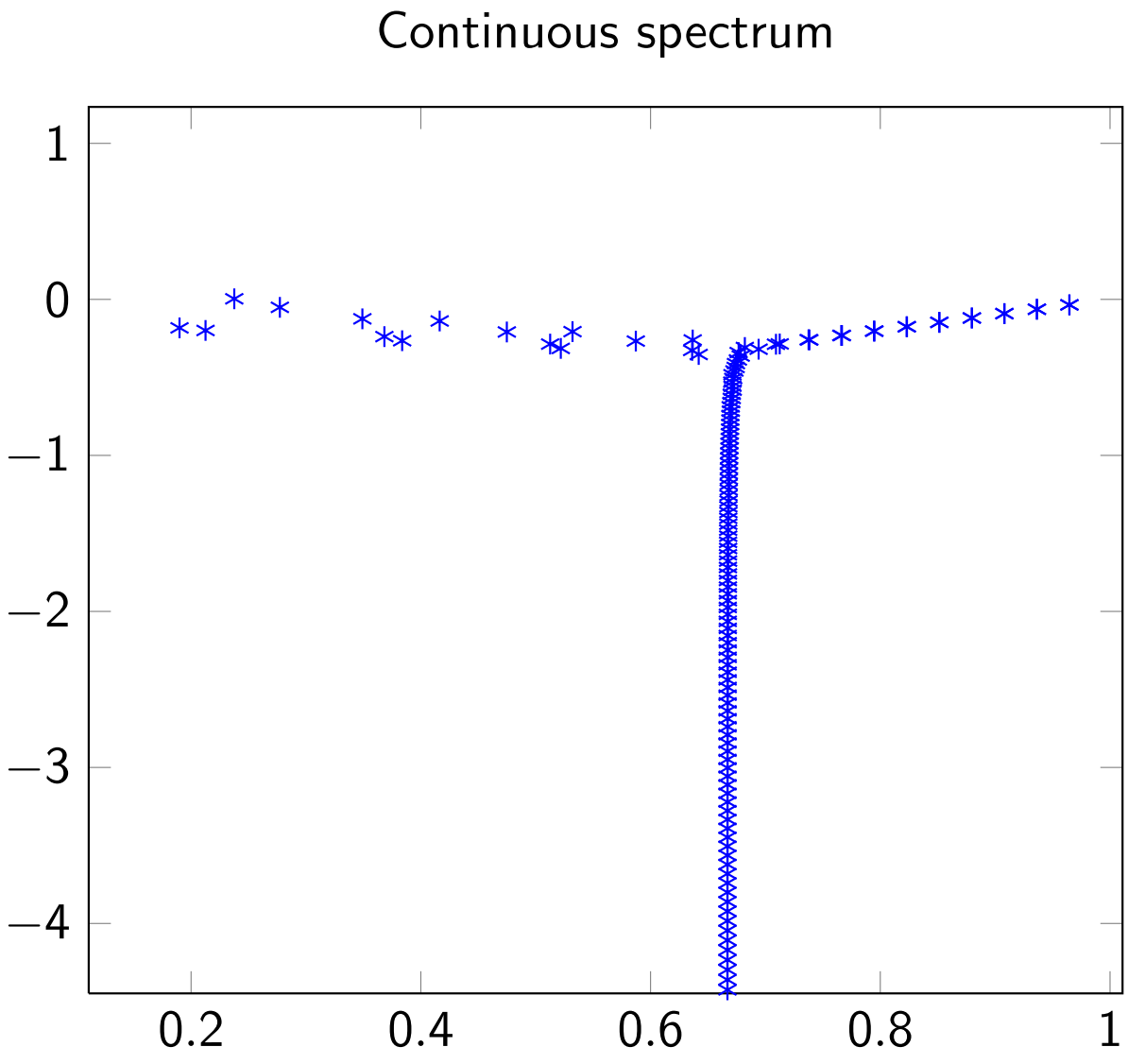
Using finite differences, we discretized with discretization step $h = 1/N$

$$A\underline{\tilde{v}} = cB\underline{\tilde{v}}$$

Where $A, B \in \mathbb{R}^{N\times N}$, $\det(A) \neq 0$, $\text{rank}(B) = N - 4$ because of B.C.
A and B are banded matrices with bandwidth respectively 5 and 3.

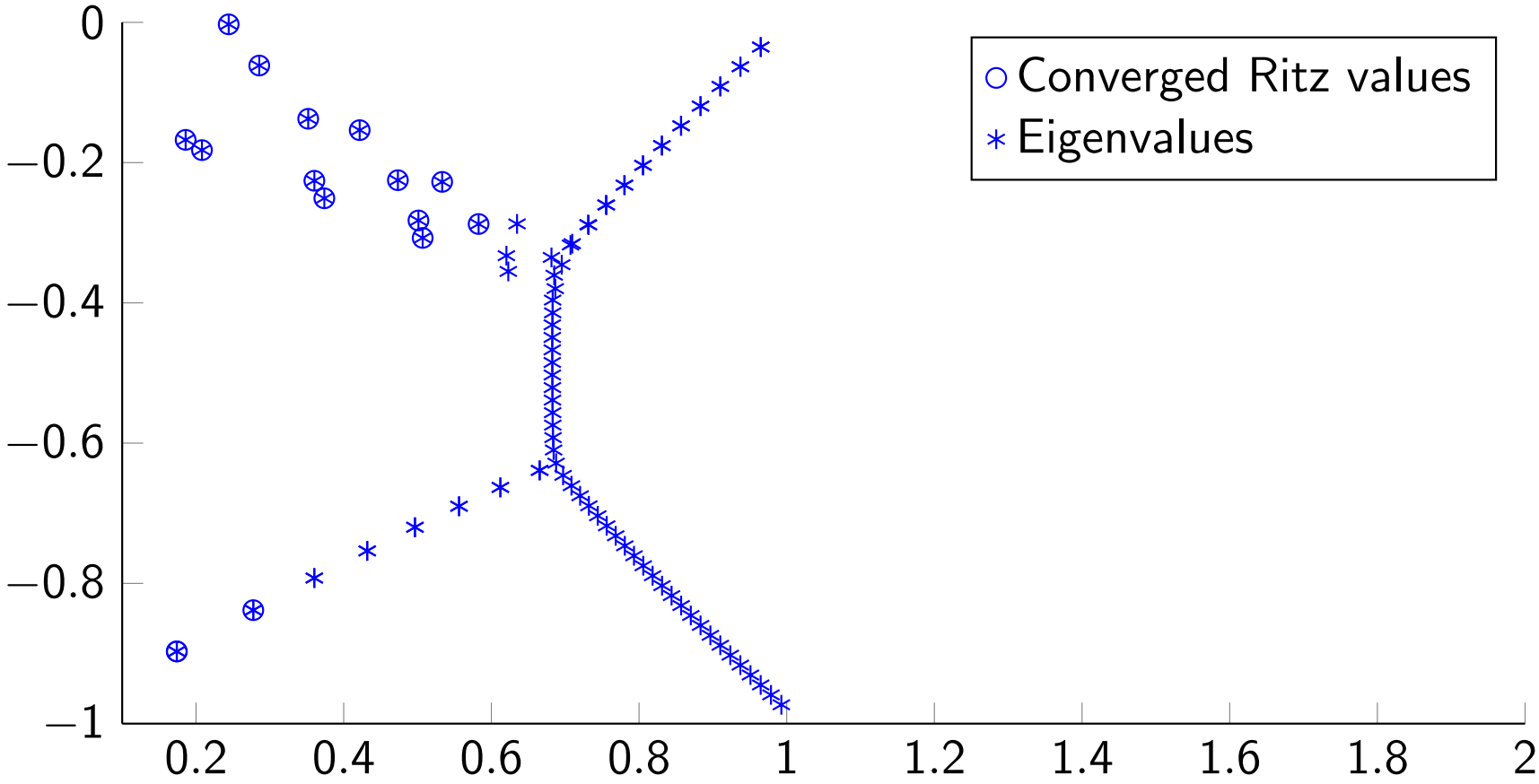
The spectrum of the continuum problem has a branch structure, in particular it looks like a Y. The task is to compute the branch connected to zero.

Numerical experimentation



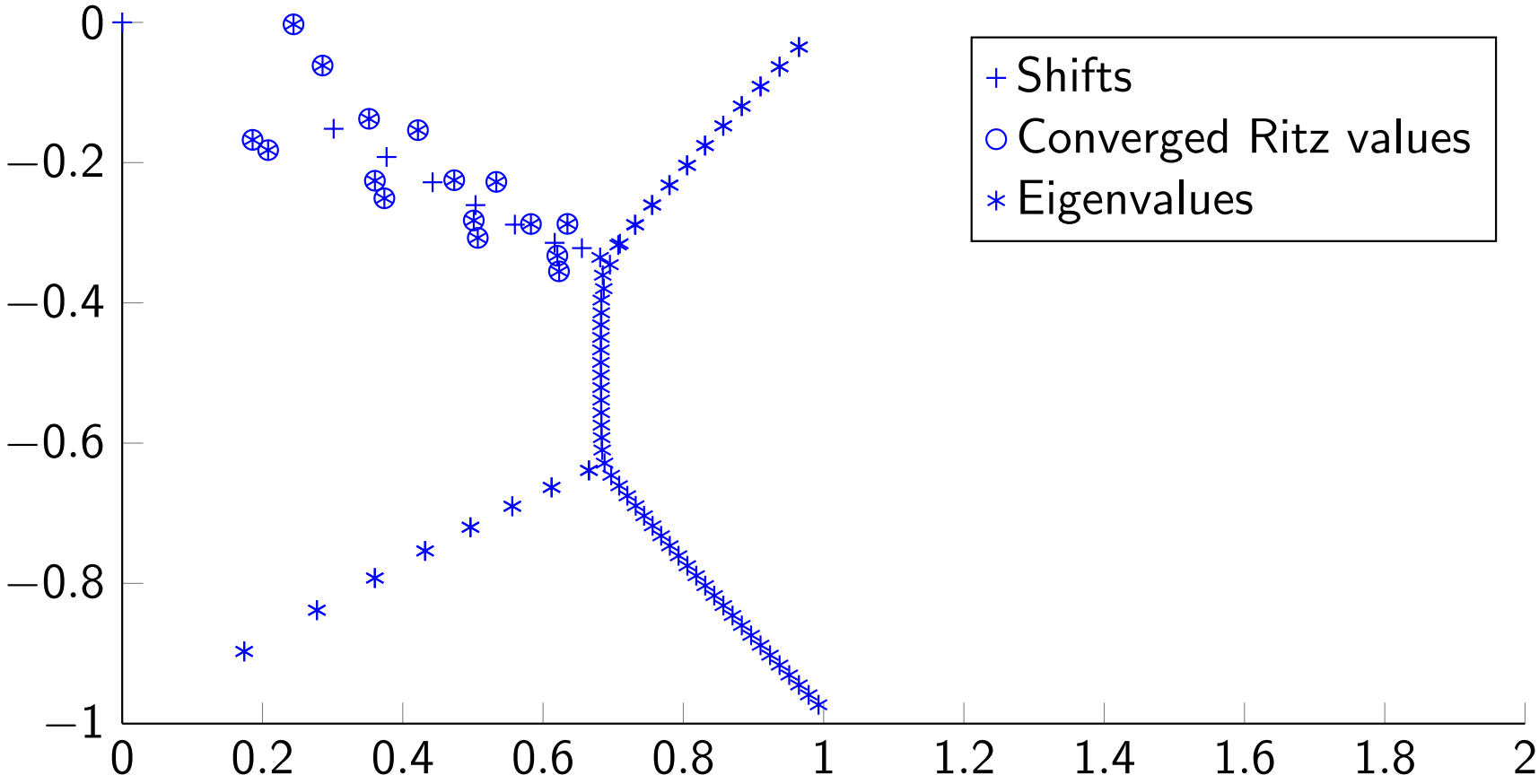
Numerical experimentation

$N = 100$, Ritz values computed with shift–invert Arnoldi.



Numerical experimentation

$N = 100$, Ritz values computed with Rational Krylov.



Outline

- Arnoldi (and its variants) for linear eigenproblems
- Rational Krylov algorithm for linear eigenproblems
- Applications of Rational Krylov algorithm for nonlinear eigenproblems
  - Linearization by means of Hermite interpolation
  - Iterative projection methods

Outline

- Arnoldi (and its variants) for linear eigenproblems
- Rational Krylov algorithm for linear eigenproblems
- Applications of Rational Krylov algorithm for nonlinear eigenproblems
  - Linearization by means of Hermite interpolations
  - Iterative projection methods

Nonlinear eigenvalue problem and linearization

Nonlinear eigenvalue problem (NLEP)

Given a nonlinear application

$$A(\cdot) : \mathbb{C} \to \mathbb{C}^{n \times n}$$

the task is to compute $(\lambda, x) \in \mathbb{C} \times \mathbb{C}^{n \times n}$ such that $A(\lambda)x = 0$ with $\lambda \in \Omega \subset \mathbb{C}$

Linearization

Given a nonlinear eigenvalue problem defined by $A(\lambda)$, the application $\widetilde{A}(\lambda)$ is a linearization if it defines a linear eigenvalue problem such that its eigenvalues (in Ω) are a good estimation of the eigenvalues (in Ω) of the original problem.

We can every time express the nonlinear eigenvalue problem as

$$A(\lambda) = \sum_{i=1}^{m} f_i(\lambda) B_i \qquad \begin{array}{l} B_i \in \mathbb{C}^{n \times n} \\ f_i : \mathbb{C} \to \mathbb{C} \end{array}$$

Nonlinear eigenvalue problem and linearization

Nonlinear eigenvalue problem (NLEP)

Given a nonlinear application

$$A(\cdot) : \mathbb{C} \to \mathbb{C}^{n \times n}$$

the task is to compute $(\lambda, x) \in \mathbb{C} \times \mathbb{C}^{n \times n}$ such that $A(\lambda)x = 0$ with $\lambda \in \Omega \subset \mathbb{C}$

Linearization

Given a nonlinear eigenvalue problem defined by $A(\lambda)$, the application $\widetilde{A}(\lambda)$ is a linearization if it defines a linear eigenvalue problem such that its eigenvalues (in Ω) are a good estimation of the eigenvalues (in Ω) of the original problem.

We can every time express the nonlinear eigenvalue problem as

$$A(\lambda) = \sum_{i=1}^{m} f_i(\lambda) B_i \qquad \begin{array}{l} B_i \in \mathbb{C}^{n \times n} \\ f_i : \mathbb{C} \to \mathbb{C} \end{array}$$

Nonlinear eigenvalue problem and linearization

Nonlinear eigenvalue problem (NLEP)

Given a nonlinear application

$$A(\cdot) : \mathbb{C} \to \mathbb{C}^{n \times n}$$

the task is to compute $(\lambda, x) \in \mathbb{C} \times \mathbb{C}^{n \times n}$ such that $A(\lambda)x = 0$ with $\lambda \in \Omega \subset \mathbb{C}$

Linearization

Given a nonlinear eigenvalue problem defined by $A(\lambda)$, the application $\widetilde{A}(\lambda)$ is a linearization if it defines a linear eigenvalue problem such that its eigenvalues (in Ω) are a good estimation of the eigenvalues (in Ω) of the original problem.

We can every time express the nonlinear eigenvalue problem as

$$A(\lambda) = \sum_{i=1}^{m} f_i(\lambda) B_i \qquad \begin{array}{l} B_i \in \mathbb{C}^{n \times n} \\ f_i : \mathbb{C} \to \mathbb{C} \end{array}$$

Linearization by means of Hermite interpolation

Consider the NLEP defined by

$$A(\lambda) = \sum_{i=1}^{m} f_i(\lambda) B_i$$

and select a set of points $\sigma_0, \ldots, \sigma_N \in \Omega$ (repetitions are allowed)

$$f_j(\lambda) \xrightarrow[\text{interpolation}]{\text{Hermite}} \sum_{i=0}^{N} \alpha_{i,j} n_i(\lambda)$$

then we can approximate the NLEP with a PEP defined by

$$P_N(\lambda) = \sum_{i=0}^{N} n_i(\lambda) A_i \qquad \text{where} \qquad A_i = \sum_{j=1}^{m} \alpha_{i,j} B_j$$

Linearization by means of Hermite interpolation

Theorem (Companion-type linearization)

The pair $(\lambda, x) \neq 0$ is an eigenpair of the PEP if and only if $\mathcal{A}_N y_N = \lambda \mathcal{B}_N y_N$ where

$$\mathcal{A}_N := \begin{pmatrix} A_0 & A_1 & A_2 & \dots & A_N \\ \sigma_0 I & I & & & \\ & \sigma_1 I & I & & \\ & & \ddots & \ddots & \\ & & & \sigma_{N-1} I & I \end{pmatrix}, \mathcal{B}_N := \begin{pmatrix} 0 & & & & \\ I & 0 & & & \\ & I & 0 & & \\ & & \ddots & \ddots & \\ & & & I & 0 \end{pmatrix}, y_N := \begin{pmatrix} x \\ n_1(\lambda)x \\ n_2(\lambda)x \\ n_3(\lambda)x \\ \vdots \\ n_N(\lambda)x \end{pmatrix}$$

Advantages

- Since $A_i = \sum_{j=1}^{m} \alpha_{i,j} B_j$, it is not needed to store A_i, it is sufficient to store the interpolation coefficients $\alpha_{i,j}$.
- If it is needed to add an interpolation point, we just need to one can just compute (implicitly) A_{N+1} and add a column and a row to the linearization matrices.
- Only the coefficients $\alpha_{i,j}$ are stored, all the other matrices are implicitly built.

Linearization by means of Hermite interpolation

Theorem (Companion-type linearization)

The pair $(\lambda, x) \neq 0$ is an eigenpair of the PEP if and only if $\mathcal{A}_N y_N = \lambda \mathcal{B}_N y_N$ where

$$\mathcal{A}_N := \begin{pmatrix} A_0 & A_1 & A_2 & \dots & A_N \\ \sigma_0 I & I & & & \\ & \sigma_1 I & I & & \\ & & \ddots & \ddots & \\ & & & \sigma_{N-1} I & I \end{pmatrix}, \mathcal{B}_N := \begin{pmatrix} 0 & & & & \\ I & 0 & & & \\ & I & 0 & & \\ & & \ddots & \ddots & \\ & & & I & 0 \end{pmatrix}, y_N := \begin{pmatrix} x \\ n_1(\lambda)x \\ n_2(\lambda)x \\ n_3(\lambda)x \\ \vdots \\ n_N(\lambda)x \end{pmatrix}$$

Advantages

- Since $A_i = \sum_{j=1}^{m} \alpha_{i,j} B_j$, it is not needed to store A_i, it is sufficient to store the interpolation coefficients $\alpha_{i,j}$.
- If it is needed to add an interpolation point, we just need to one can just compute (implicitly) A_{N+1} and add a column and a row to the linearization matrices.
- Only the coefficients $\alpha_{i,j}$ are stored, all the other matrices are implicitly built.

Rational Krylov algorithm to solve the linearized problem

Lemma

Consider the linear problem defined by the linearization $(\mathcal{A}_N, \mathcal{B}_N)$*, apply the rational Krylov algorithm by using as shifts the interpolation points and*

$$v_1 := vec\left(v_1^{[1]}, 0, \ldots, 0\right), \quad v_1 \in \mathbb{C}^{(N+1)n}, \quad v_1^{[1]} \in \mathbb{C}^n$$

as starting vector. Then at the j*-th step of the rational Krylov algorithm the vectors of the Arnoldi sequence have the following structure*

$$v_k = vec\left(v_k^{[1]}, v_k^{[2]}, \ldots, v_k^{[j]}, 0, \ldots, 0\right), \quad for \quad k \leq j \leq N,$$

where $v_k^{[i]} \in \mathbb{C}^n$ *for* $i = 1, \ldots, j$.

Building the basis of the Krylov subspace

Size of linearization: $N = 8$ (blocks)
Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$v_1 = \begin{pmatrix} v_1^{[1]} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Building the basis of the Krylov subspace

Size of linearization: $N = 8$ (blocks)
Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$v_1 = \begin{pmatrix} v_1^{[1]} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$v_2 = \begin{pmatrix} v_2^{[1]} & v_2^{[2]} & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Building the basis of the Krylov subspace

Size of linearization: $N = 8$ (blocks)

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2	
$v_1 = ($	$v_1^{[1]}$	0	0	0	0	0	0	0 $)$
$v_2 = ($	$v_2^{[1]}$	$v_2^{[2]}$	0	0	0	0	0	0 $)$
$v_3 = ($	$v_3^{[1]}$	$v_3^{[2]}$	$v_3^{[3]}$	0	0	0	0	0 $)$

Building the basis of the Krylov subspace

Size of linearization: $N = 8$ (blocks)

Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$
\begin{aligned}
v_1 &= \begin{pmatrix} v_1^{[1]} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
v_2 &= \begin{pmatrix} v_2^{[1]} & v_2^{[2]} & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
v_3 &= \begin{pmatrix} v_3^{[1]} & v_3^{[2]} & v_3^{[3]} & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
v_4 &= \begin{pmatrix} v_4^{[1]} & v_4^{[2]} & v_4^{[3]} & v_4^{[4]} & 0 & 0 & 0 & 0 \end{pmatrix}
\end{aligned}
$$

Building the basis of Krylov subspace

Size of linearization: $N = 8$ (blocks)
Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$
\begin{aligned}
w_1 &= (\; w_1^{[1]} \; w_1^{[2]} \; w_1^{[3]} \; w_1^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_2 &= (\; w_2^{[1]} \; w_2^{[2]} \; w_2^{[3]} \; w_2^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_3 &= (\; w_3^{[1]} \; w_3^{[2]} \; w_3^{[3]} \; w_3^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_4 &= (\; w_4^{[1]} \; w_4^{[2]} \; w_4^{[3]} \; w_4^{[4]} \; 0 \; 0 \; 0 \; 0 \;)
\end{aligned}
$$

Building the basis of Krylov subspace

Size of linearization: $N = 8$ (blocks)

Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$
\begin{aligned}
w_1 &= (\; w_1^{[1]} \; w_1^{[2]} \; w_1^{[3]} \; w_1^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_2 &= (\; w_2^{[1]} \; w_2^{[2]} \; w_2^{[3]} \; w_2^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_3 &= (\; w_3^{[1]} \; w_3^{[2]} \; w_3^{[3]} \; w_3^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_4 &= (\; w_4^{[1]} \; w_4^{[2]} \; w_4^{[3]} \; w_4^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_5 &= (\; w_5^{[1]} \; w_5^{[2]} \; w_5^{[3]} \; w_5^{[4]} \; w_5^{[5]} \; 0 \; 0 \; 0 \;)
\end{aligned}
$$

Building the basis of Krylov subspace

Size of linearization: $N = 8$ (blocks)

Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$
\begin{aligned}
w_1 &= (\; w_1^{[1]} \; w_1^{[2]} \; w_1^{[3]} \; w_1^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_2 &= (\; w_2^{[1]} \; w_2^{[2]} \; w_2^{[3]} \; w_2^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_3 &= (\; w_3^{[1]} \; w_3^{[2]} \; w_3^{[3]} \; w_3^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_4 &= (\; w_4^{[1]} \; w_4^{[2]} \; w_4^{[3]} \; w_4^{[4]} \; 0 \; 0 \; 0 \; 0 \;) \\
w_5 &= (\; w_5^{[1]} \; w_5^{[2]} \; w_5^{[3]} \; w_5^{[4]} \; w_5^{[5]} \; 0 \; 0 \; 0 \;) \\
w_6 &= (\; w_6^{[1]} \; w_6^{[2]} \; w_6^{[3]} \; w_6^{[4]} \; w_6^{[5]} \; w_6^{[6]} \; 0 \; 0 \;)
\end{aligned}
$$

Building the basis of Krylov subspace

Size of linearization: $N = 8$ (blocks)
Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$
\begin{aligned}
z_1 &= (\; z_1^{[1]} \; z_1^{[2]} \; z_1^{[3]} \; z_1^{[4]} \; z_1^{[5]} \; z_1^{[6]} \; 0 \; 0 \;) \\
z_2 &= (\; z_2^{[1]} \; z_2^{[2]} \; z_2^{[3]} \; z_2^{[4]} \; z_2^{[5]} \; z_2^{[6]} \; 0 \; 0 \;) \\
z_3 &= (\; z_3^{[1]} \; z_3^{[2]} \; z_3^{[3]} \; z_3^{[4]} \; z_3^{[5]} \; z_3^{[6]} \; 0 \; 0 \;) \\
z_4 &= (\; z_4^{[1]} \; z_4^{[2]} \; z_4^{[3]} \; z_4^{[4]} \; z_4^{[5]} \; z_4^{[6]} \; 0 \; 0 \;) \\
z_5 &= (\; z_5^{[1]} \; z_5^{[2]} \; z_5^{[3]} \; z_5^{[4]} \; z_5^{[5]} \; z_5^{[6]} \; 0 \; 0 \;) \\
z_6 &= (\; z_6^{[1]} \; z_6^{[2]} \; z_6^{[3]} \; z_6^{[4]} \; z_6^{[5]} \; z_6^{[6]} \; 0 \; 0 \;)
\end{aligned}
$$

Building the basis of Krylov subspace

Size of linearization: $N = 8$ (blocks)

Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$
\begin{aligned}
z_1 &= (\; z_1^{[1]} \; z_1^{[2]} \; z_1^{[3]} \; z_1^{[4]} \; z_1^{[5]} \; z_1^{[6]} \; 0 \; 0 \;) \\
z_2 &= (\; z_2^{[1]} \; z_2^{[2]} \; z_2^{[3]} \; z_2^{[4]} \; z_2^{[5]} \; z_2^{[6]} \; 0 \; 0 \;) \\
z_3 &= (\; z_3^{[1]} \; z_3^{[2]} \; z_3^{[3]} \; z_3^{[4]} \; z_3^{[5]} \; z_3^{[6]} \; 0 \; 0 \;) \\
z_4 &= (\; z_4^{[1]} \; z_4^{[2]} \; z_4^{[3]} \; z_4^{[4]} \; z_4^{[5]} \; z_4^{[6]} \; 0 \; 0 \;) \\
z_5 &= (\; z_5^{[1]} \; z_5^{[2]} \; z_5^{[3]} \; z_5^{[4]} \; z_5^{[5]} \; z_5^{[6]} \; 0 \; 0 \;) \\
z_6 &= (\; z_6^{[1]} \; z_6^{[2]} \; z_6^{[3]} \; z_6^{[4]} \; z_6^{[5]} \; z_6^{[6]} \; 0 \; 0 \;) \\
z_7 &= (\; z_7^{[1]} \; z_7^{[2]} \; z_7^{[3]} \; z_7^{[4]} \; z_7^{[5]} \; z_7^{[6]} \; z_7^{[7]} \; 0 \;)
\end{aligned}
$$

Building the basis of Krylov subspace

Size of linearization: $N = 8$ (blocks)

Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$
\begin{aligned}
z_1 &= (\; z_1^{[1]} \; z_1^{[2]} \; z_1^{[3]} \; z_1^{[4]} \; z_1^{[5]} \; z_1^{[6]} \; 0 \; 0 \;) \\
z_2 &= (\; z_2^{[1]} \; z_2^{[2]} \; z_2^{[3]} \; z_2^{[4]} \; z_2^{[5]} \; z_2^{[6]} \; 0 \; 0 \;) \\
z_3 &= (\; z_3^{[1]} \; z_3^{[2]} \; z_3^{[3]} \; z_3^{[4]} \; z_3^{[5]} \; z_3^{[6]} \; 0 \; 0 \;) \\
z_4 &= (\; z_4^{[1]} \; z_4^{[2]} \; z_4^{[3]} \; z_4^{[4]} \; z_4^{[5]} \; z_4^{[6]} \; 0 \; 0 \;) \\
z_5 &= (\; z_5^{[1]} \; z_5^{[2]} \; z_5^{[3]} \; z_5^{[4]} \; z_5^{[5]} \; z_5^{[6]} \; 0 \; 0 \;) \\
z_6 &= (\; z_6^{[1]} \; z_6^{[2]} \; z_6^{[3]} \; z_6^{[4]} \; z_6^{[5]} \; z_6^{[6]} \; 0 \; 0 \;) \\
z_7 &= (\; z_7^{[1]} \; z_7^{[2]} \; z_7^{[3]} \; z_7^{[4]} \; z_7^{[5]} \; z_7^{[6]} \; z_7^{[7]} \; 0 \;) \\
z_8 &= (\; z_8^{[1]} \; z_8^{[2]} \; z_8^{[3]} \; z_8^{[4]} \; z_8^{[5]} \; z_8^{[6]} \; z_8^{[7]} \; z_8^{[8]} \;)
\end{aligned}
$$

Rational Krylov algorithm to solve the linearized problem

Lemma

At each iteration j of the rational Krylov algorithm, only the top-left parts of the matrices $\mathcal{A}_N - \sigma_j \mathcal{B}_N$ are used to compute the nonzero top parts $\tilde{v}_{j+1}$ of the vectors v_{j+1}, i.e.,

$$(\mathcal{A}_j - \sigma_j \mathcal{B}_j)\tilde{v}_{j+1} = \mathcal{B}_j \tilde{v}_j,$$

where

$$\tilde{v}_{j+1} = vec\left(v_{j+1}^{[1]}, v_{j+1}^{[2]}, \ldots, v_{j+1}^{[j+1]}\right),$$

and

$$\tilde{v}_j = vec\left(v_j^{[1]}, v_j^{[2]}, \ldots, v_j^{[j]}, 0\right),$$

Rational Krylov algorithm to solve the linearized problem

Lemma

The linear system $(\mathcal{A}_j - \sigma_j \mathcal{B}_j)\tilde{v}_{j+1} = \mathcal{B}_j \tilde{v}_j$ *can be efficiently solved by using the following equations*

$$A(\sigma_j) v_{j+1}^{[1]} = y_0^{(j)},$$

where

$$y_0^{(j)} = -\sum_{i=1}^{j} A_j \left(v_j^{[i]} + \sum_{k=1}^{i-1} \left(\prod_{l=k}^{i-1} \mu_l^{(j)} \right) v_j^{[k]} \right),$$

and

$$\begin{aligned} v_{j+1}^{[2]} &= v_j^{[1]} + \mu_0^{(j)} v_{j+1}^{[1]}, \\ v_{j+1}^{[3]} &= v_j^{[2]} + \mu_1^{(j)} v_{j+1}^{[2]}, \\ &\vdots, \\ v_{j+1}^{[j+1]} &= v_j^{[j]} + \mu_{j-1}^{(j)} v_{j+1}^{[j]}. \end{aligned}$$

HIRK (Hermite Interpolation Rational Krylov Method)

1: Choose the shift σ_0 and starting vector v_1.
2: **for** $j = 1, \ldots, m$ **do**
3: EXPANSION PHASE.
4: Choose the shift σ_j.
5: Compute the next divided difference: A_j.
6: Expand $\mathcal{A}_j$, $\mathcal{B}_j$ and V_j.
7: RATIONAL KRYLOV STEP
8: **if** $\sigma_{j-1} \neq \sigma_j$ **then**
9: Change basis $V_j \to \tilde{V}_j$ and matrix $H_{j,j-1} \to \tilde{H}_{j,j-1}$
(according to the Rational Krylov algorithm)
such that the Arnoldi sequence becomes

$$(\mathcal{A}_j - \sigma_j \mathcal{B}_j)^{-1} \mathcal{B}_j \tilde{V}_j = \tilde{H}_{j,j-1} V_{j-1}.$$

10: **end if**
11: Compute the next vector of the sequence:

$$r = (\mathcal{A}_j - \sigma_j \mathcal{B}_j)^{-1} \mathcal{B}_j v_j,$$

$$r = v - V_j h_j, \qquad \text{where} \quad h_j = V_j^H r \qquad \text{orthogonalization},$$

$$v_{j+1} = r / h_{j+1,j}, \qquad \text{where} \quad h_{j+1,j} = \|r\| \qquad \text{normalization}.$$

12: Compute the eigenpair (θ_i, y_i) for $i = 1, \ldots, j$ of $H_{j,j-1}$ and then the Ritz pairs $(\theta_i, V_j y_i)$.
13: Test the convergence for the NLEP.
14: **end for**

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$v_1 = ($ $v_1^{[1]}$ $)$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$v_1 = ($ $v_1^{[1]}$ 0 $)$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$v_1 = \begin{pmatrix} v_1^{[1]} & 0 \end{pmatrix}$$

$$v_2 = \begin{pmatrix} v_2^{[1]} & v_2^{[2]} \end{pmatrix}$$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts: σ_0 σ_0 σ_0 σ_1 σ_1 σ_2 σ_2

$$v_1 = \begin{pmatrix} v_1^{[1]} & 0 & 0 \end{pmatrix}$$

$$v_2 = \begin{pmatrix} v_2^{[1]} & v_2^{[2]} & 0 \end{pmatrix}$$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2

$$v_1 = \begin{pmatrix} v_1^{[1]} & 0 & 0 \end{pmatrix}$$

$$v_2 = \begin{pmatrix} v_2^{[1]} & v_2^{[2]} & 0 \end{pmatrix}$$

$$v_3 = \begin{pmatrix} v_3^{[1]} & v_3^{[2]} & v_3^{[3]} \end{pmatrix}$$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2

$$
\begin{aligned}
v_1 &= (\quad v_1^{[1]} \quad 0 \quad 0 \quad 0 \quad) \\
v_2 &= (\quad v_2^{[1]} \quad v_2^{[2]} \quad 0 \quad 0 \quad) \\
v_3 &= (\quad v_3^{[1]} \quad v_3^{[2]} \quad v_3^{[3]} \quad 0 \quad)
\end{aligned}
$$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2

$$
\begin{aligned}
v_1 &= (\quad v_1^{[1]} \quad 0 \quad 0 \quad 0 \quad) \\
v_2 &= (\quad v_2^{[1]} \quad v_2^{[2]} \quad 0 \quad 0 \quad) \\
v_3 &= (\quad v_3^{[1]} \quad v_3^{[2]} \quad v_3^{[3]} \quad 0 \quad) \\
v_4 &= (\quad v_4^{[1]} \quad v_4^{[2]} \quad v_4^{[3]} \quad v_4^{[4]} \quad)
\end{aligned}
$$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2

$$
\begin{aligned}
w_1 &= (\quad w_1^{[1]} \quad w_1^{[2]} \quad w_3^{[3]} \quad w_1^{[4]} \quad) \\
w_2 &= (\quad w_2^{[1]} \quad w_2^{[2]} \quad w_2^{[3]} \quad w_2^{[4]} \quad) \\
w_3 &= (\quad w_3^{[1]} \quad w_3^{[2]} \quad w_3^{[3]} \quad w_3^{[4]} \quad) \\
w_4 &= (\quad w_4^{[1]} \quad w_4^{[2]} \quad w_4^{[3]} \quad w_4^{[4]} \quad)
\end{aligned}
$$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2
$w_1 = ($	$w_1^{[1]}$	$w_1^{[2]}$	$w_3^{[3]}$	$w_1^{[4]}$	0	$)$	
$w_2 = ($	$w_2^{[1]}$	$w_2^{[2]}$	$w_2^{[3]}$	$w_2^{[4]}$	0	$)$	
$w_3 = ($	$w_3^{[1]}$	$w_3^{[2]}$	$w_3^{[3]}$	$w_3^{[4]}$	0	$)$	
$w_4 = ($	$w_4^{[1]}$	$w_4^{[2]}$	$w_4^{[3]}$	$w_4^{[4]}$	0	$)$	

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2

$$
\begin{aligned}
w_1 &= (\quad w_1^{[1]} \quad w_1^{[2]} \quad w_3^{[3]} \quad w_1^{[4]} \quad 0 \quad) \\
w_2 &= (\quad w_2^{[1]} \quad w_2^{[2]} \quad w_2^{[3]} \quad w_2^{[4]} \quad 0 \quad) \\
w_3 &= (\quad w_3^{[1]} \quad w_3^{[2]} \quad w_3^{[3]} \quad w_3^{[4]} \quad 0 \quad) \\
w_4 &= (\quad w_4^{[1]} \quad w_4^{[2]} \quad w_4^{[3]} \quad w_4^{[4]} \quad 0 \quad) \\
w_5 &= (\quad w_5^{[1]} \quad w_5^{[2]} \quad w_5^{[3]} \quad w_5^{[4]} \quad w_5^{[5]} \quad)
\end{aligned}
$$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2
$w_1 = ($	$w_1^{[1]}$	$w_1^{[2]}$	$w_3^{[3]}$	$w_1^{[4]}$	0	0	$)$
$w_2 = ($	$w_2^{[1]}$	$w_2^{[2]}$	$w_2^{[3]}$	$w_2^{[4]}$	0	0	$)$
$w_3 = ($	$w_3^{[1]}$	$w_3^{[2]}$	$w_3^{[3]}$	$w_3^{[4]}$	0	0	$)$
$w_4 = ($	$w_4^{[1]}$	$w_4^{[2]}$	$w_4^{[3]}$	$w_4^{[4]}$	0	0	$)$
$w_5 = ($	$w_5^{[1]}$	$w_5^{[2]}$	$w_5^{[3]}$	$w_5^{[4]}$	$w_5^{[5]}$	0	$)$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2
$w_1 = ($	$w_1^{[1]}$	$w_1^{[2]}$	$w_3^{[3]}$	$w_1^{[4]}$	0	0	$)$
$w_2 = ($	$w_2^{[1]}$	$w_2^{[2]}$	$w_2^{[3]}$	$w_2^{[4]}$	0	0	$)$
$w_3 = ($	$w_3^{[1]}$	$w_3^{[2]}$	$w_3^{[3]}$	$w_3^{[4]}$	0	0	$)$
$w_4 = ($	$w_4^{[1]}$	$w_4^{[2]}$	$w_4^{[3]}$	$w_4^{[4]}$	0	0	$)$
$w_5 = ($	$w_5^{[1]}$	$w_5^{[2]}$	$w_5^{[3]}$	$w_5^{[4]}$	$w_5^{[5]}$	0	$)$
$w_6 = ($	$w_6^{[1]}$	$w_6^{[2]}$	$w_6^{[3]}$	$w_6^{[4]}$	$w_6^{[5]}$	$w_6^{[6]}$	$)$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2
$z_1 = ($	$z_1^{[1]}$	$z_1^{[2]}$	$z_1^{[3]}$	$z_1^{[4]}$	$z_1^{[5]}$	$z_1^{[6]}$	$)$
$z_2 = ($	$z_2^{[1]}$	$z_2^{[2]}$	$z_2^{[3]}$	$z_2^{[4]}$	$z_2^{[5]}$	$z_2^{[6]}$	$)$
$z_3 = ($	$z_3^{[1]}$	$z_3^{[2]}$	$z_3^{[3]}$	$z_3^{[4]}$	$z_3^{[5]}$	$z_3^{[6]}$	$)$
$z_4 = ($	$z_4^{[1]}$	$z_4^{[2]}$	$z_4^{[3]}$	$z_4^{[4]}$	$z_4^{[5]}$	$z_4^{[6]}$	$)$
$z_5 = ($	$z_5^{[1]}$	$z_5^{[2]}$	$z_5^{[3]}$	$z_5^{[4]}$	$z_5^{[5]}$	$z_5^{[6]}$	$)$
$z_6 = ($	$z_6^{[1]}$	$z_6^{[2]}$	$z_6^{[3]}$	$z_6^{[4]}$	$z_6^{[5]}$	$z_6^{[6]}$	$)$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2
$z_1 = ($	$z_1^{[1]}$	$z_1^{[2]}$	$z_3^{[3]}$	$z_1^{[4]}$	$z_1^{[5]}$	$z_1^{[6]}$	0 $)$
$z_2 = ($	$z_2^{[1]}$	$z_2^{[2]}$	$z_2^{[3]}$	$z_2^{[4]}$	$z_2^{[5]}$	$z_2^{[6]}$	0 $)$
$z_3 = ($	$z_3^{[1]}$	$z_3^{[2]}$	$z_3^{[3]}$	$z_3^{[4]}$	$z_3^{[5]}$	$z_3^{[6]}$	0 $)$
$z_4 = ($	$z_4^{[1]}$	$z_4^{[2]}$	$z_4^{[3]}$	$z_4^{[4]}$	$z_4^{[5]}$	$z_4^{[6]}$	0 $)$
$z_5 = ($	$z_5^{[1]}$	$z_5^{[2]}$	$z_5^{[3]}$	$z_5^{[4]}$	$z_5^{[5]}$	$z_5^{[6]}$	0 $)$
$z_6 = ($	$z_6^{[1]}$	$z_6^{[2]}$	$z_6^{[3]}$	$z_6^{[4]}$	$z_6^{[5]}$	$z_6^{[6]}$	0 $)$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2
$z_1 = ($	$z_1^{[1]}$	$z_1^{[2]}$	$z_3^{[3]}$	$z_1^{[4]}$	$z_1^{[5]}$	$z_1^{[6]}$	0 $)$
$z_2 = ($	$z_2^{[1]}$	$z_2^{[2]}$	$z_2^{[3]}$	$z_2^{[4]}$	$z_2^{[5]}$	$z_2^{[6]}$	0 $)$
$z_3 = ($	$z_3^{[1]}$	$z_3^{[2]}$	$z_3^{[3]}$	$z_3^{[4]}$	$z_3^{[5]}$	$z_3^{[6]}$	0 $)$
$z_4 = ($	$z_4^{[1]}$	$z_4^{[2]}$	$z_4^{[3]}$	$z_4^{[4]}$	$z_4^{[5]}$	$z_4^{[6]}$	0 $)$
$z_5 = ($	$z_5^{[1]}$	$z_5^{[2]}$	$z_5^{[3]}$	$z_5^{[4]}$	$z_5^{[5]}$	$z_5^{[6]}$	0 $)$
$z_6 = ($	$z_6^{[1]}$	$z_6^{[2]}$	$z_6^{[3]}$	$z_6^{[4]}$	$z_6^{[5]}$	$z_6^{[6]}$	0 $)$
$z_7 = ($	$z_7^{[1]}$	$z_7^{[2]}$	$z_7^{[3]}$	$z_7^{[4]}$	$z_7^{[5]}$	$z_7^{[6]}$	$z_7^{[7]}$ $)$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2	
$z_1 = ($	$z_1^{[1]}$	$z_1^{[2]}$	$z_3^{[3]}$	$z_1^{[4]}$	$z_1^{[5]}$	$z_1^{[6]}$	0	0 $)$
$z_2 = ($	$z_2^{[1]}$	$z_2^{[2]}$	$z_2^{[3]}$	$z_2^{[4]}$	$z_2^{[5]}$	$z_2^{[6]}$	0	0 $)$
$z_3 = ($	$z_3^{[1]}$	$z_3^{[2]}$	$z_3^{[3]}$	$z_3^{[4]}$	$z_3^{[5]}$	$z_3^{[6]}$	0	0 $)$
$z_4 = ($	$z_4^{[1]}$	$z_4^{[2]}$	$z_4^{[3]}$	$z_4^{[4]}$	$z_4^{[5]}$	$z_4^{[6]}$	0	0 $)$
$z_5 = ($	$z_5^{[1]}$	$z_5^{[2]}$	$z_5^{[3]}$	$z_5^{[4]}$	$z_5^{[5]}$	$z_5^{[6]}$	0	0 $)$
$z_6 = ($	$z_6^{[1]}$	$z_6^{[2]}$	$z_6^{[3]}$	$z_6^{[4]}$	$z_6^{[5]}$	$z_6^{[6]}$	0	0 $)$
$z_7 = ($	$z_7^{[1]}$	$z_7^{[2]}$	$z_7^{[3]}$	$z_7^{[4]}$	$z_7^{[5]}$	$z_7^{[6]}$	$z_7^{[7]}$	0 $)$

Building the basis of Krylov subspace
Execution of the algorithm

Nodes/shifts:	σ_0	σ_0	σ_0	σ_1	σ_1	σ_2	σ_2	
$z_1 = ($	$z_1^{[1]}$	$z_1^{[2]}$	$z_3^{[3]}$	$z_1^{[4]}$	$z_1^{[5]}$	$z_1^{[6]}$	0	0 $)$
$z_2 = ($	$z_2^{[1]}$	$z_2^{[2]}$	$z_2^{[3]}$	$z_2^{[4]}$	$z_2^{[5]}$	$z_2^{[6]}$	0	0 $)$
$z_3 = ($	$z_3^{[1]}$	$z_3^{[2]}$	$z_3^{[3]}$	$z_3^{[4]}$	$z_3^{[5]}$	$z_3^{[6]}$	0	0 $)$
$z_4 = ($	$z_4^{[1]}$	$z_4^{[2]}$	$z_4^{[3]}$	$z_4^{[4]}$	$z_4^{[5]}$	$z_4^{[6]}$	0	0 $)$
$z_5 = ($	$z_5^{[1]}$	$z_5^{[2]}$	$z_5^{[3]}$	$z_5^{[4]}$	$z_5^{[5]}$	$z_5^{[6]}$	0	0 $)$
$z_6 = ($	$z_6^{[1]}$	$z_6^{[2]}$	$z_6^{[3]}$	$z_6^{[4]}$	$z_6^{[5]}$	$z_6^{[6]}$	0	0 $)$
$z_7 = ($	$z_7^{[1]}$	$z_7^{[2]}$	$z_7^{[3]}$	$z_7^{[4]}$	$z_7^{[5]}$	$z_7^{[6]}$	$z_7^{[7]}$	0 $)$
$z_8 = ($	$z_8^{[1]}$	$z_8^{[2]}$	$z_8^{[3]}$	$z_8^{[4]}$	$z_8^{[5]}$	$z_8^{[6]}$	$z_8^{[7]}$	$z_8^{[8]}$ $)$

Hermite Interpolation Rational Krylov Method

Comments

- At every step it is solved a system of the size of the original NLEP and not of the size of the linarization.
- The computation of the interpolation coefficients is numerically unstable. These coefficients must be computed semianalitically.
- Applying this method to a NLEP is like to solve a linear eigenvalue problem of infinite size.
- The bottleneck of the algorithm is the Gram–Schmidt process.
- At every step, the vectors of the basis of Krylov space get longer.
- Exploiting the low rank structure of the matrix coefficients can speedup the algorithm.

Outline

- Arnoldi (and its variants) for linear eigenproblems
- Rational Krylov algorithm for linear eigenproblems
- Applications of Rational Krylov algorithm for nonlinear eigenproblems
  - Linearization by means of Hermite interpolations
  - Iterative projection methods

Nonlinear Rational Krylov

Definition (Generalized Arnoldi's sequence)

Given a pole $\sigma \in \mathbb{C}$ and a sequence of shifts $\lambda_1, \ldots, \lambda_m$ it holds

$$A(\sigma)^{-1} A(\lambda_m) V_m = V_{m+1} H_{m+1,m}$$

Generation of the sequence

$$A(\sigma)^{-1} A(\lambda_{j-1}) V_{j-1} = V_j H_{j,j-1} \xrightarrow[\text{interpolation}]{\text{linear}} A(\sigma)^{-1} A(\lambda_j) V_j = V_{j+1} \overline{H}_{j+1,j}.$$

Nonlinear Rational Krylov

Definition (Generalized Arnoldi's sequence)

Given a pole $\sigma \in \mathbb{C}$ and a sequence of shifts $\lambda_1, \ldots, \lambda_m$ it holds

$$A(\sigma)^{-1} A(\lambda_m) V_m = V_{m+1} H_{m+1,m}$$

Generation of the sequence

$$A(\sigma)^{-1} A(\lambda_{j-1}) V_{j-1} = V_j H_{j,j-1} \xrightarrow[\text{interpolation}]{\text{linear}} A(\sigma)^{-1} A(\lambda_j) V_j = V_{j+1} \overline{H}_{j+1,j}.$$

Nonlinear Rational Krylov

Observation

- With a linear Lagrange–interpolation between λ_j and σ we get the linearized problem

$$A(\lambda) = \frac{\lambda - \lambda_j}{\sigma - \lambda_j} A(\sigma) + \frac{\lambda - \sigma}{\lambda_j - \sigma} A(\lambda_j).$$

- If (θ, x) is such that

$$A(\sigma)^{-1} A(\lambda_j) x = \theta x$$

then (λ_{j+1}, x) is an eigenpair or the linearized problem, where

$$\lambda_{j+1} = \lambda_j + \frac{\theta}{1 - \theta}(\lambda_j - \sigma).$$

The closer θ to 0 the closer λ_{j+1} to λ_j.

Nonlinear Rational Krylov

Nonlinear Rational Krylov algorithm (Preliminary version)

1: Choose a starting vector v_1
2: **for** $j = 1, \ldots,$ till convergence **do**
3: Compute the Arnoldi sequence $A(\lambda_j)V_j = A(\sigma)V_{j+1}H_{j+1,j}$
4: Compute the smallest eigenpairs (θ, s) of $H_{j,j}$
5: $\lambda_{j+1} = \lambda_j + \frac{\theta}{1-\theta}(\lambda_j - \sigma)$
6: $H_{j+1,j} = \frac{1}{1-\theta}H_{j+1,j} - \frac{\theta}{1-\theta}I_{j+1,j}$
7: **end for**

It turns out that this algorithm does not work well.

Nonlinear Rational Krylov

Nonlinear Rational Krylov algorithm (Preliminary version)

1: Choose a starting vector v_1
2: **for** $j = 1, \ldots,$ till convergence **do**
3: Compute the Arnoldi sequence $A(\lambda_j)V_j = A(\sigma)V_{j+1}H_{j+1,j}$
4: Compute the smallest eigenpairs (θ, s) of $H_{j,j}$
5: $\lambda_{j+1} = \lambda_j + \frac{\theta}{1-\theta}(\lambda_j - \sigma)$
6: $H_{j+1,j} = \frac{1}{1-\theta}H_{j+1,j} - \frac{\theta}{1-\theta}I_{j+1,j}$
7: **end for**

It turns out that this algorithm does not work well.

Nonlinear Rational Krylov

Proposition

It holds

$$A(\sigma)^{-1}A(\lambda_{j+1})V_j - V_j H_{j,j} = A(\sigma)^{-1}A(\lambda_{j+1})V_j s\ e_j^H.$$

Observation

In the linear case it holds

$$A(\sigma)^{-1}A(\lambda_{j+1})V_j s = s_j h_{j+1,j} v_{j+1}$$

that is, the residual is orthogonal to V_m. This property does not hold in the nonlinear case. We can introduce *INNER ITERATIONS* to enforce it.

Nonlinear Rational Krylov

Proposition

It holds

$$A(\sigma)^{-1}A(\lambda_{j+1})V_j - V_j H_{j,j} = A(\sigma)^{-1}A(\lambda_{j+1})V_j s\ e_j^H.$$

Observation

In the linear case it holds

$$A(\sigma)^{-1}A(\lambda_{j+1})V_j s = s_j h_{j+1,j} v_{j+1}$$

that is, the residual is orthogonal to V_m. This property does not hold in the nonlinear case. We can introduce *INNER ITERATIONS* to enforce it.

NLRK

1: Choose a starting vector v_1 with $\|v_1\| = 1$, a starting shift λ_1 and a pole σ and set $j = 1$.
2: OUTER ITERATION
3: Set $h_j = 0; s = e_j = (0, \ldots, 0, 1)^H \in \mathbb{R}^j; x = v_j;$
4: Compute $r = A(\sigma)^{-1}A(\lambda)x$ and $k_j = V_j^H r$
5: **while** $\|k_J\| > ResTol$ **do**
6: INNER ITERATION
7: Orthogonalize $r = r - Vk_j$
8: Set $h_j = h_j + s_j^{-1}k_j$
9: Compute the smallest eigenpair (θ, s) of $H_{j,j}$
10: $x = V_j s$
11: Update $\lambda = \lambda + \frac{\theta}{1-\theta}(\lambda - \theta)$
12: Update $H_{j,j} = \frac{1}{1-\theta}H_{j,j} - \frac{\theta}{1-\theta}I$
13: Compute $r = A(\sigma)^{-1}A(\lambda)x$ and $k_j = V_j^H r$
14: **end while**
15: Compute $h_{j+1,j} = \|r\|/s_j$
16: **if** $|h_{j+1,j}s_j| > EigTol$ **then**
17: $v_{j+1} = r/\|r\|; j = j + 1;$ GOTO 3
18: **end if**
19: Store (θ, x) as eigenpair
20: If more eigenvalues are requested, choose next θ and s, and GOTO 10

Numerical experimentation

GUN problem

This is a large-scale NLEP that models a radio frequency gun cavity and is of the form

$$F(\lambda)x = \left(K - \lambda M + i\sqrt{\lambda - \sigma_1^2}W_1 + i\sqrt{\lambda - \sigma_2^2}W_2 \right) x = 0$$

Where $M, K, W_1, W_2 \in \mathbb{R}^{9956\times 9956}$ are real symmetric, K is positive semidefinite, and M is positive definite. The domain of interest is

$$\Omega = \{\lambda \in \mathbb{C} \text{ such that } |\lambda - \mu| \leq \gamma \quad \text{and} \quad Im(\lambda) \geq 0\}.$$

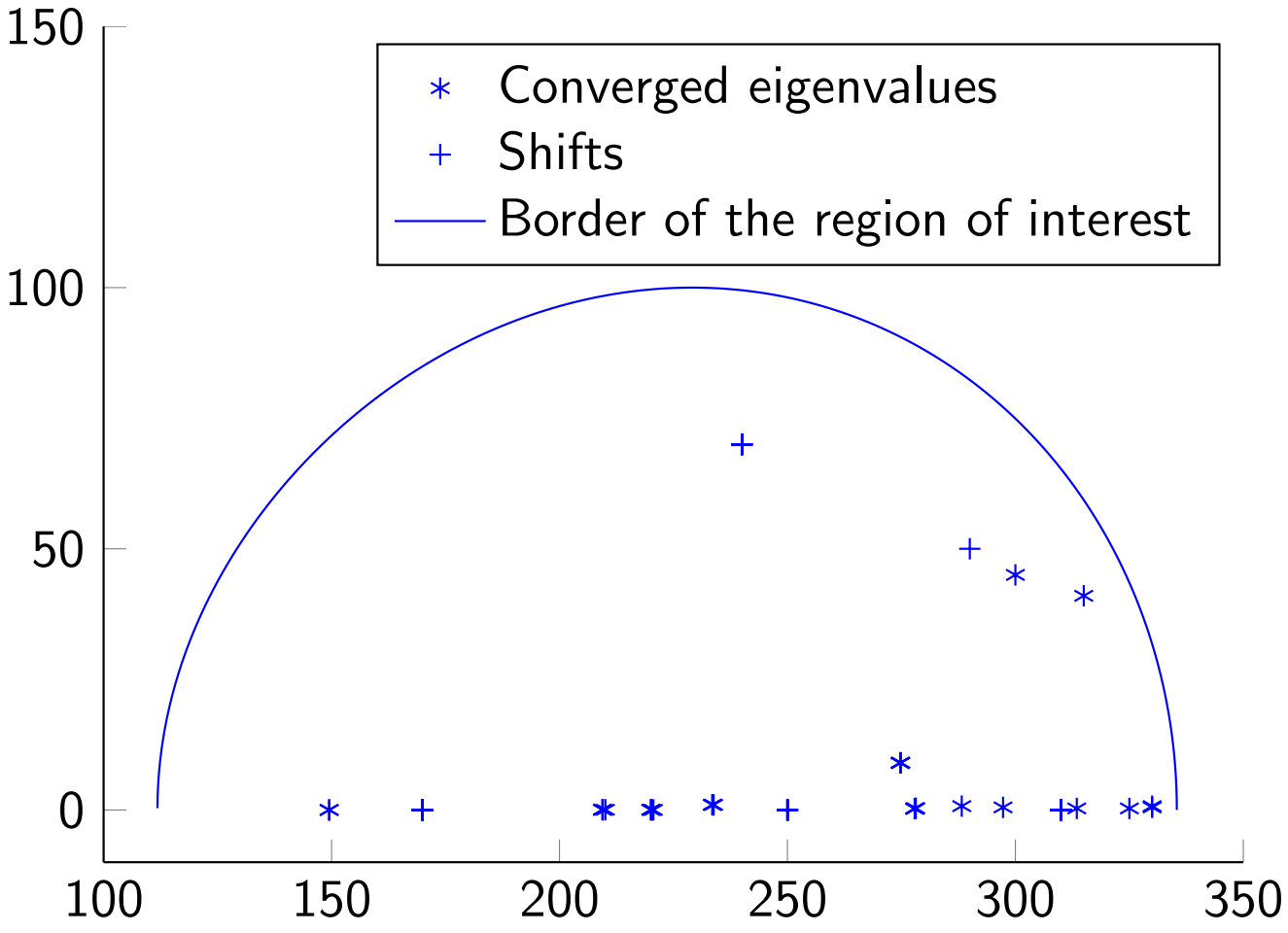
The parameters are set to $\sigma_1 = 0, \sigma_2 = 108.8774, \gamma = 50000$ and $\mu = 62500$.

Before solving the problem we applied shift and rescaling in order to transform Ω into the upper part of the unit circle.

Numerical experimentation: HIRK

- NLRK diverges
- HIRK succeeds to compute eigenvalues

Eigenvalues of the gun problem are computed with 60 iterations. The same node is used 12 times.

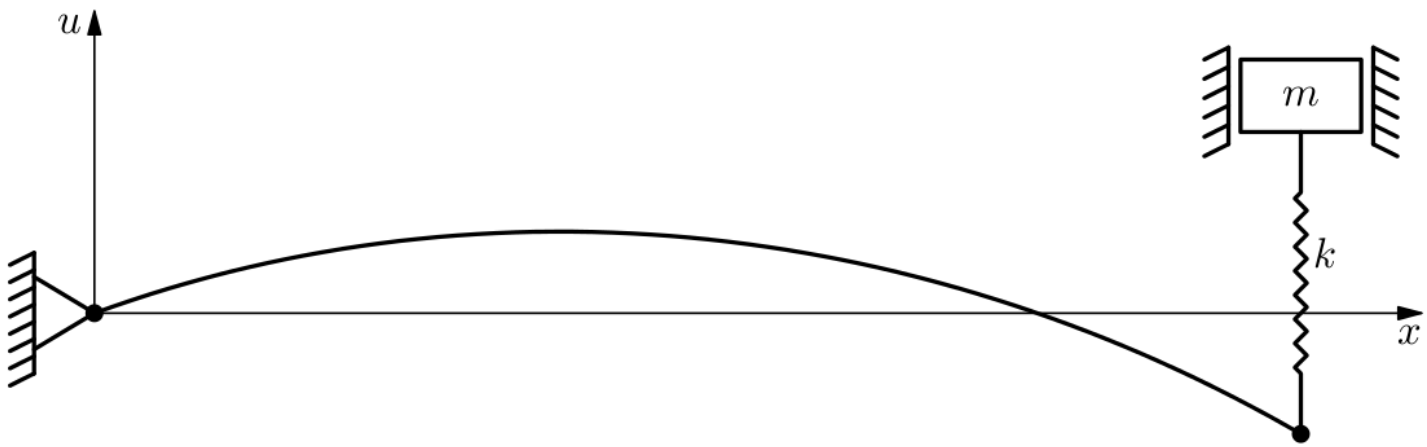


Numerical experimentation

Vibrating string with elastically attached mass

Consider the system of a limp string of unit length, which is clamped at one end. The other end is free but has a mass m attached to it via an elastic spring of stiffness k. The eigenvibrations of the string are governed by the eigenvalue problem

$$\begin{cases} -u''(x) = \lambda u(x) \\ u(0) = 0 \\ u'(1) + k\frac{\lambda}{\lambda - k/m}u(1) = 0 \end{cases}$$



Numerical experimentation

Eigenvalues of the continuum problem

With easy computations, we found that the eigenvalues are the solution of the equation

$$\tan(\sqrt{\lambda}) = \frac{1}{m\lambda} - \frac{\sqrt{\lambda}}{k}$$

Discrete problem

Discretizing the problem by means of the finite element method using $P1$ elements we arrive at the nonlinear eigenproblem

$$A - \lambda B + k\frac{\lambda}{\lambda - k/m} C = 0,$$

$$A = \frac{1}{h}\begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & 2 & -1 \\ & & -1 & 1 \end{pmatrix}, \quad B = \frac{h}{6}\begin{pmatrix} 4 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & 4 & 1 \\ & & 1 & 2 \end{pmatrix}, \quad C = e_n e_n^H.$$

Numerical experimentation

Eigenvalues of the continuum problem

With easy computations, we found that the eigenvalues are the solution of the equation

$$\tan(\sqrt{\lambda}) = \frac{1}{m\lambda} - \frac{\sqrt{\lambda}}{k}$$

Discrete problem

Discretizing the problem by means of the finite element method using $P1$ elements we arrive at the nonlinear eigenproblem

$$A - \lambda B + k\frac{\lambda}{\lambda - k/m}C = 0,$$

$$A = \frac{1}{h}\begin{pmatrix} 2 & -1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}, \quad B = \frac{h}{6}\begin{pmatrix} 4 & 1 & & & \\ 1 & \ddots & \ddots & & \\ & \ddots & & 4 & 1 \\ & & & 1 & 2 \end{pmatrix}, \quad C = e_n e_n^H.$$

Numerical experimentation: NLRK

Task

Compute the second smallest eigenvalue λ_2

Se set $EigTol = 10^{-6}$ and $ResTol = 10^{-6}$.

For $m = 1$ and $k = 0.01$ we have $\lambda_2 \simeq 2.4874$.

N	$\lvert\lambda_2 - \tilde{\lambda}_2\rvert$	Outer iterations	Average of inner iterations
100	10^{-3}	5	2
10000	10^{-5}	6	2

For $m = 1$ and $k = 0.1$ we have $\lambda_2 \simeq 2.6679$.

N	$\lvert\lambda_2 - \tilde{\lambda}_2\rvert$	Outer iterations	Average of inner iterations
100	10^{-2}	5	3
10000	10^{-3}	6	3

For $m = 1$ and $k = 1$ NLRK diverges.

HIRK succeeds to compute λ_2 but it is slow. On the other hand it works also for $m = 1$ and $k = 1$.

Numerical experimentation

Fluid-solid structure interaction

The study of free vibrations of a tube bundle immersed in a slightly compressible (under a few simplifications) leads to the following continuum eigenproblem.
Find $\lambda \in \mathbb{R}$ and $u \in H^1(\Omega_0)$ such that for every $v \in H^1(\Omega_0)$

$$c^2 \int_{\Omega_0} \nabla u \cdot \nabla v dx = \lambda \int_{\Omega_0} uv dx + \sum_{j=1^k} \frac{\lambda \rho_0}{k_j - \lambda m_j} \int_{\Gamma_j} u n ds \cdot \int_{\Gamma_j} v n ds$$

All the constants in the above problem are set equal to 1.

Discrete problem

After discretization by means of finite elements we obtain

$$A(\lambda)x = -Ax + \lambda Bx + \frac{\lambda}{1-\lambda} Cx = 0$$

where C collects the contributions of all tubes. A, B, and C are symmetric matrices, A and C are positive semidefinite, and B is positive definite

Numerical experimentation

Fluid-solid structure interaction

The study of free vibrations of a tube bundle immersed in a slightly compressible (under a few simplifications) leads to the following continuum eigenproblem. Find $\lambda \in \mathbb{R}$ and $u \in H^1(\Omega_0)$ such that for every $v \in H^1(\Omega_0)$

$$c^2 \int_{\Omega_0} \nabla u \cdot \nabla v dx = \lambda \int_{\Omega_0} uv dx + \sum_{j=1^k} \frac{\lambda \rho_0}{k_j - \lambda m_j} \int_{\Gamma_j} und s \cdot \int_{\Gamma_j} vnds$$

All the constants in the above problem are set equal to 1.

Discrete problem

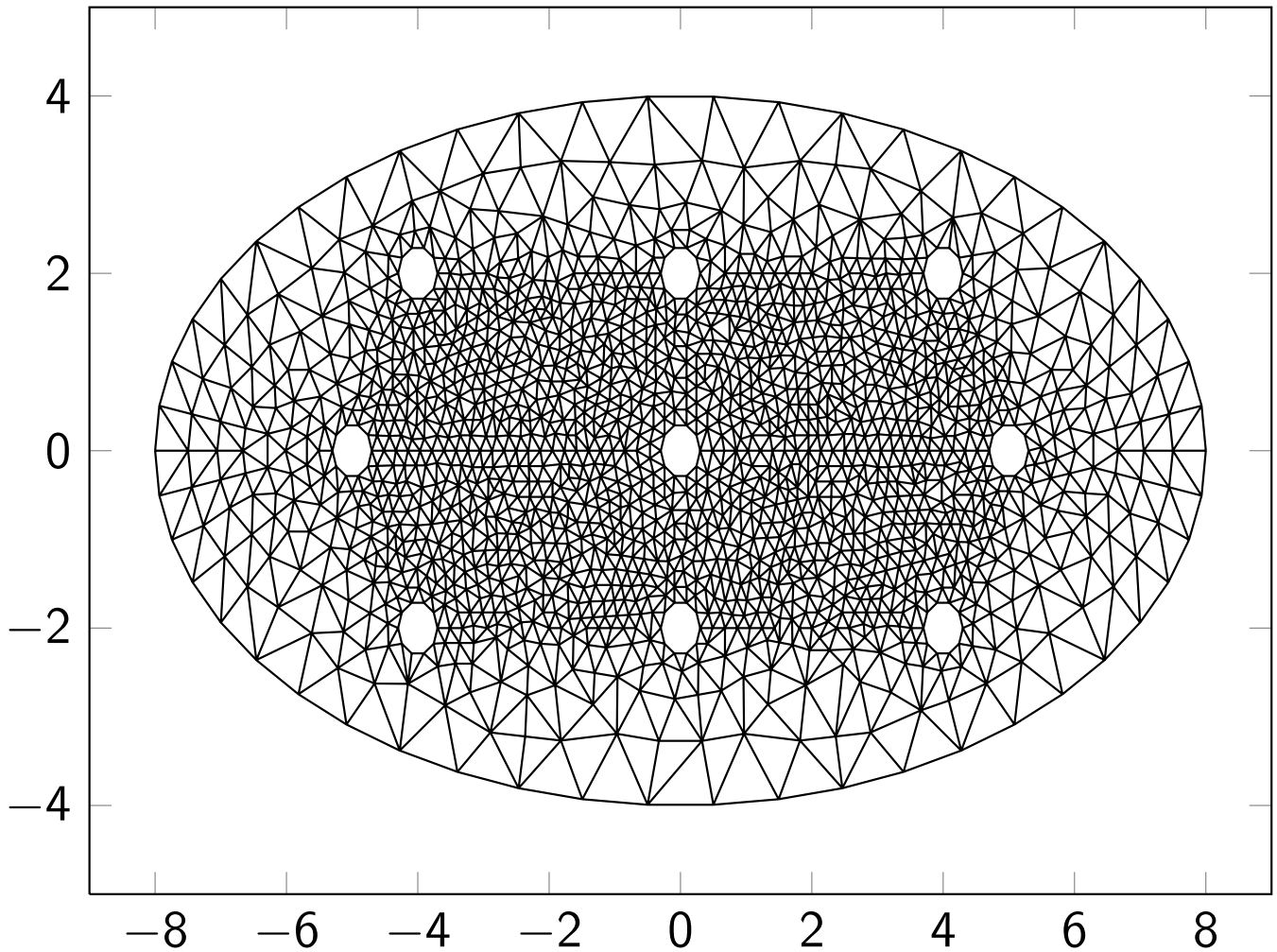
After discretization by means of finite elements we obtain

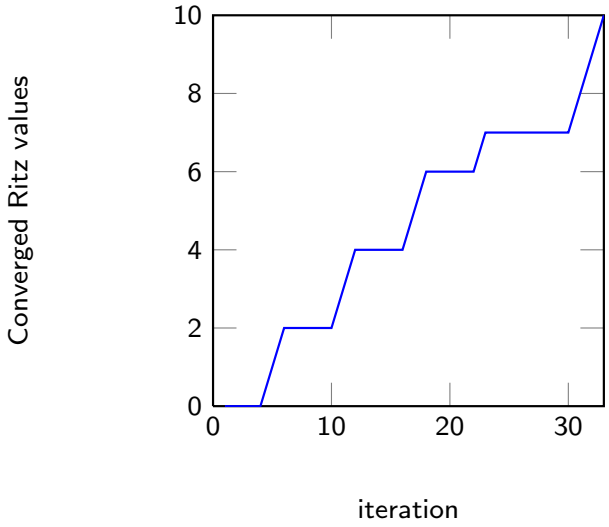
$$A(\lambda)x = -Ax + \lambda Bx + \frac{\lambda}{1-\lambda} Cx = 0$$

where C collects the contributions of all tubes. A, B, and C are symmetric matrices, A and C are positive semidefinite, and B is positive definite

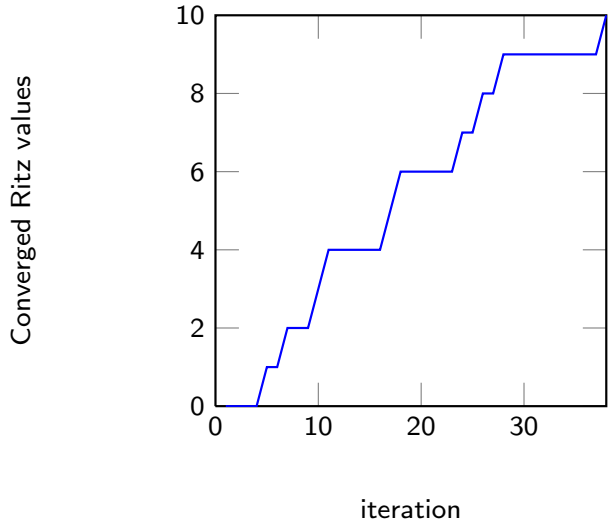
Numerical experimentation

In our setting there are 9 tubes. We discretized the problem with FreeFem++ using P1 triangular elements. Example of discretization of domain with FreeFem++

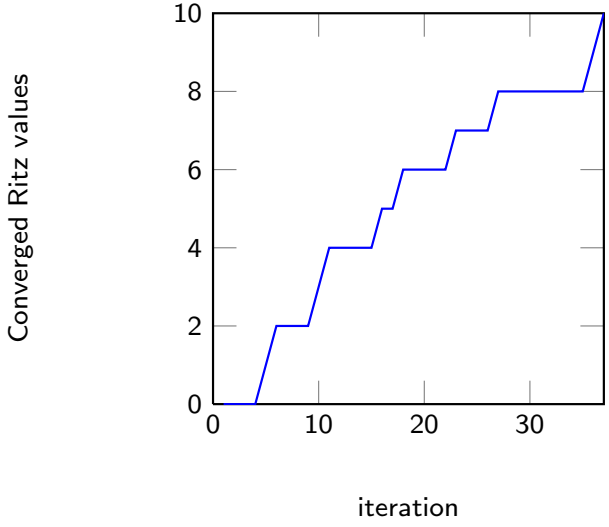




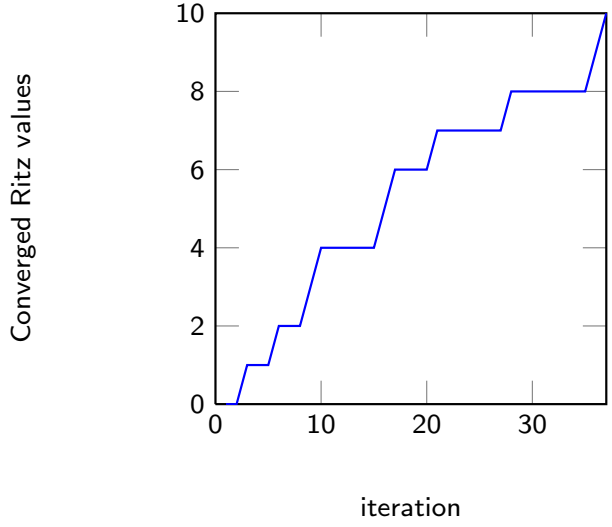
(a) $n = 50,\ m = 10,\ N = 1636$



(b) $n = 100,\ m = 10,\ N = 2156$



(c) $n = 200,\ m = 10,\ N = 3277$



(d) $n = 400,\ m = 10,\ N = 5604$

Thank you for your attention.