An Algorithm for the Generalized Symmetric Tridiagonal Eigenvalue Problem

Kuiyuam Li, Tien-Yien Li, Zhonggang Zeng

Giulio Masetti

Università di Pisa
Corso Metodi di Approssimazione 2012-2013

August 6, 2013

Generalized Eigenvalue Problem

Def

$T, S \in \mathbb{R}^{n\times n}$. We call (T, S) *pencil*.

We consider *only* $\boxed{\begin{array}{l}\text{symmetric}\\ \text{and}\\ \text{tridiagonal}\end{array}}$ pencil.

Generalized Eigenvalue Problem

Def

$T, S \in \mathbb{R}^{n \times n}$. We call (T, S) *pencil*.

We consider *only* $\boxed{\begin{array}{l}\text{symmetric}\\ \text{and}\\ \text{tridiagonal}\end{array}}$ pencil.

S is *positive definite*.

Generalized Eigenvalue Problem

Def

$T, S \in \mathbb{R}^{n\times n}$. We call (T, S) *pencil.*

We consider *only* $\boxed{\begin{array}{l}\text{symmetric}\\ \text{and}\\ \text{tridiagonal}\end{array}}$ pencil.

S is *positive definite*.

Def (Problem)

Find $\lambda \in \mathbb{R}$ such that $Tv = \lambda S v$, with $v \in \mathbb{C}^n$.
T, S symmetric implies $\lambda \in \mathbb{R}$.

Algorithm philosophy

We find zeros of the polynomial equation

$$\mathcal{F}_{(T,S)}(\lambda) = \det(T - \lambda S) = 0$$

using an iterative method, living on real line.

Brainstorming

We want:	Fast and secure iterative method. Starting points for our method. Scalability.
We have:	Laguerre's method. Cuppen's divide and conquer method. Symmetric tridiagonal matrices.
We add:	Unreducible condition. Dynamic programming (Bottom-up). Efficient matrix storing.

Rapid tour

We want:	Fast and secure iterative method. Starting points for our method. Scalability.
We have:	Laguerre's method. Cuppen's divide and conquer method. Symmetric tridiagonal matrices.
We add:	Unreducible condition. Dynamic programming (Bottom-up). Efficient matrix storing.

Unreducible pencil

Def (as in [1])

(T, S) is an *unreducible pencil* if $t_{i,i+1}^2 + s_{i,i+1}^2 \neq 0$ for $i = 1, 2, \ldots, n-1$.

Unreducible pencil

Def (as in [1])

(T, S) is an *unreducible pencil* if $t^2_{i,i+1} + s^2_{i,i+1} \neq 0$ for $i = 1, 2, \ldots, n-1$.

exemplum gratie:

Bad

$$\boxed{\begin{array}{l} T = I, S = 0 \\ T = I, S = I \end{array}}$$

Good

$$\boxed{\begin{array}{l} T = I, S = trid(-1, 2, -1) \\ T = trid(-1, 2, -1), S = trid(-1, 2, -1) \\ T = trid(rnd_{sub}, rnd_{diag}, rnd_{sub}), S = I, \\ \text{with } rnd_{sub} \text{ random number} \neq 0. \end{array}}$$

Matrix storin

$$T = trid(sub, diag, super)$$

But T is symmetric, so $sub = super$. We define and use

```
integer, parameter :: dp = kind(1.d0)
real(dp), dimension(1:n,0:1) :: T, S
```

with $T(:,0) = diag$ and $T(:,1) = super$.

Oss

We don't use T(1,1) and S(1,1).

Fast and secure iterative method

We want:	Fast and secure iterative method. Starting points for our method. Scalability.
We have:	Laguerre's method. Cuppen's divide and conquer method. Symmetric tridiagonal matrices.
We add:	Unreducible condition. Dynamic programming (Bottom-up). Efficient matrix storing.

Fast and secure iterative method

$\mathcal{F}_{T,S}(\lambda)$ is a polynomial with only real zeros; we call them

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

where we count with multiplicity.
If λ_m and λ_{m+1} are simple zeros (mlt=1), then we consider the quadric

$$g_u(X) = (x - X)^2 \sum_{i=1}^{n} \frac{(u - \lambda_i)^2}{(x - \lambda_i)^2} - (u - X)^2$$

if $\boxed{u \neq x}$ then $g_u(x) < 0$ and $g_u(\lambda_m) > 0$.
So we have two sign change.
Bolzano's Theorem tell us that there are two zeros of g_u between λ_m and λ_{m+1}. We call them X_-, X_+.

Fast and secure iterative method

$$\lambda_m < X_- < x < X_+ < \lambda_{m+1}$$

We have one freedom: the u parameter.
Calling $\hat{X}_- = min_u X_-$ and $\hat{X}_+ = max_u X_+$ we can obtain

$$\lambda_m \approx \hat{X}_- < x < \hat{X}_+ \approx \lambda_{m+1}$$

and with algebraic manipulations:

$$\hat{X}_-, \hat{X}_+ = L_\pm(x) = x + \frac{n}{-\frac{f'}{f} \pm \sqrt{(n-1)[(n-1)(-\frac{f'}{f}) - n\frac{f''}{f}]}}$$

with $\frac{f'}{f} = \frac{(\mathcal{F}_{T,S}(\lambda))'}{\mathcal{F}_{T,S}(\lambda)}$.

Fast and secure iterative method

So we need $\frac{f'}{f}$ and $\frac{f''}{f}$.

Fast and secure iterative method

So we need $\frac{f'}{f}$ and $\frac{f''}{f}$.
This is one of the most important aspect of our calculation.

Fast and secure iterative method

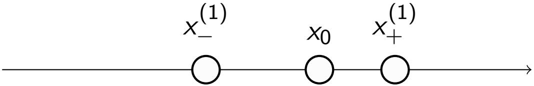
So we need $\frac{f'}{f}$ and $\frac{f''}{f}$.
This is one of the most important aspect of our calculation.
We will see that "only" with the symmetric tridiagonal condition we can have derivatives of determinats.

Fast and secure iterative method

It's clear how we use $\lambda_m \approx \hat{X}_- < x < \hat{X}_+ \approx \lambda_{m+1}$:

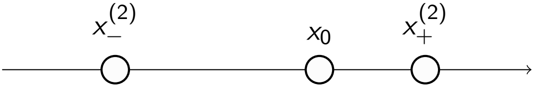
Fast and secure iterative method

It's clear how we use $\lambda_m \approx \hat{X}_- < x < \hat{X}_+ \approx \lambda_{m+1}$:



Fast and secure iterative method

It's clear how we use $\lambda_m \approx \hat{X}_- < x < \hat{X}_+ \approx \lambda_{m+1}$:



Fast and secure iterative method

It's clear how we use $\lambda_m \approx \hat{X}_- < x < \hat{X}_+ \approx \lambda_{m+1}$:

Fast and secure iterative method

Def (Laguerre's iteration)

If $mlt(\lambda_m) = mlt(\lambda_{m+1}) = 1$ then

$$
\begin{aligned}
x_+^{(k)} &= L_+^k(x) = L_+(L_+(\dots(x_0))) \\
x_-^{(k)} &= L_-^k(x) = L_-(L_-(\dots(x_0)))
\end{aligned}
$$

else we have a similar expression.

Fast and secure iterative method

Def (Laguerre's iteration)

If $mlt(\lambda_m) = mlt(\lambda_{m+1}) = 1$ then

$$x_+^{(k)} = L_+^k(x) = L_+(L_+(\dots(x_0)))$$
$$x_-^{(k)} = L_-^k(x) = L_-(L_-(\dots(x_0)))$$

else we have a similar expression.

We can prove that

$$\lambda_m \leftarrow \dots x_-^{(2)} < x_-^{(1)} < x_0 < x_+^{(1)} < x_+^{(2)} \cdots \rightarrow \lambda_{m+1}$$

Fast and secure iterative method

Def (Laguerre's iteration)

If $mlt(\lambda_m) = mlt(\lambda_{m+1}) = 1$ then

$$x_+^{(k)} = L_+^k(x) = L_+(L_+(\dots(x_0)))$$
$$x_-^{(k)} = L_-^k(x) = L_-(L_-(\dots(x_0)))$$

else we have a similar expression.

We can prove that

$$\lambda_m \leftarrow \dots x_-^{(2)} < x_-^{(1)} < x_0 < x_+^{(1)} < x_+^{(2)} \cdots \rightarrow \lambda_{m+1}$$

So Laguerre's method is secure.

Fast and secure iterative method

We also have un important property:

Teo

*If we choose** $\lambda_m < x_0$ *s.t.* $sign(\frac{f'(x_0)}{f(x_0)}) = sign(\lambda_m - x_0)$ *then* $\{x_-^{(k)}\}_{k=1,\ldots}$ *converge monotonically in asymptotically cubic rate to* λ_m.

---

*for $x_0 < \lambda_{m+1}$ s.t. $sign(\frac{f'(x_0)}{f(x_0)}) = sign(\lambda_m - x_0)$ we have $\{x_+^{(k)}\}_{k=1,\ldots}$ conv. mon. cubic to λ_{m+1}

Fast and secure iterative method

We also have un important property:

Teo

*If we choose** $\lambda_m < x_0$ *s.t.* $sign(\frac{f^{'}(x_0)}{f(x_0)}) = sign(\lambda_m - x_0)$ *then* $\{x_-^{(k)}\}_{k=1,\dots}$ *converge monotonically in asymptotically cubic rate to* λ_m.

So we can exactly define a $\boxed{\text{neighborood ``near'' } \lambda}$ and in it we have cubic rate convergence (much, much faster then bisection)

*for $x_0 < \lambda_{m+1}$ s.t. $sign(\frac{f^{'}(x_0)}{f(x_0)}) = sign(\lambda_m - x_0)$ we have $\{x_+^{(k)}\}_{k=1,\dots}$ conv. mon. cubic to λ_{m+1}

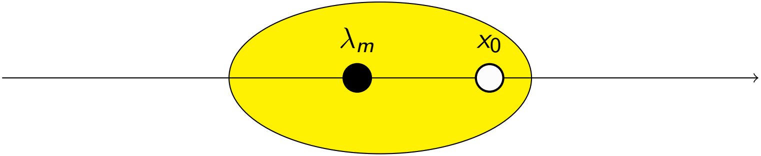
Fast and secure iterative method

We also have un important property:

Teo

*If we choose** $\lambda_m < x_0$ *s.t.* $sign(\frac{f'(x_0)}{f(x_0)}) = sign(\lambda_m - x_0)$ *then* $\{x_-^{(k)}\}_{k=1,\dots}$ *converge monotonically in asymptotically cubic rate to* λ_m.

So we can exactly define a $\boxed{\text{neighborood ``near'' } \lambda}$ and in it we have cubic rate convergence (much, much faster then bisection)



*for $x_0 < \lambda_{m+1}$ s.t. $sign(\frac{f'(x_0)}{f(x_0)}) = sign(\lambda_m - x_0)$ we have $\{x_+^{(k)}\}_{k=1,\dots}$ conv. mon. cubic to λ_{m+1}

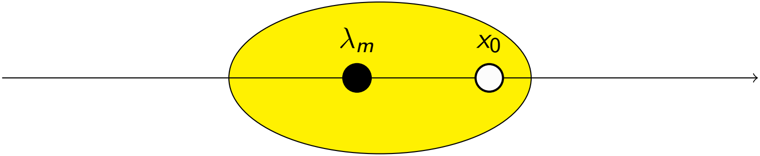
Fast and secure iterative method

We also have un important property:

Teo

*If we choose** $\lambda_m < x_0$ *s.t.* $sign(\frac{f'(x_0)}{f(x_0)}) = sign(\lambda_m - x_0)$ *then* $\{x_-^{(k)}\}_{k=1,\dots}$ *converge monotonically in asymptotically cubic rate to* λ_m.

So we can exactly define a $\boxed{\text{neighborood “near” } \lambda}$ and in it we have cubic rate convergence (much, much faster then bisection)



The Laguerre's method is fast.

*for $x_0 < \lambda_{m+1}$ s.t. $sign(\frac{f'(x_0)}{f(x_0)}) = sign(\lambda_m - x_0)$ we have $\{x_+^{(k)}\}_{k=1,\dots}$ conv. mon. cubic to λ_{m+1}

Fast and secure iterative method

It's clear that we need a powerfull method to obtain x_0 and an algorithm to estimate $mlt(\lambda_m)$.
Overstimate $mlt(\lambda_m)$ (as we can read in [2]) causes no trouble, so the most importan aspects of our calculation are:

Good x_0.

Good evaluation of $L_\pm(x)$.

Three term recurrence

For a generic $x \in \mathbb{R}$ we call $\rho_n(x) = \det(T_n - xS_n)$, and $\rho_{n-1}(x) = \det(T_{n-1} - xS_{n-1})$, with T_{n-1} leading principal submatrix.
We have

$$\rho_0 := 1\,,\ \rho_1 := t_{1,1} - xs_{1,1}$$
$$\rho_i := (t_{i,i} - xs_{i,i})\rho_{i-1} - (t_{i-1,i} - xs_{i-1,i})^2 \rho_{i-2}\,,\ i = 2, 3, \ldots, n$$

Three term recurrence

We can proove it with the *Laplace expansion* (e.g. $n = 4$)

$$T_4 - xS_4 = \begin{pmatrix} a & b & 0 & 0 \\ b & c & d & 0 \\ 0 & d & e & f \\ 0 & 0 & f & g \end{pmatrix}$$

$$det\left(T_4 - \lambda S_4\right) = g \begin{bmatrix} a & b & 0 \\ b & c & d \\ 0 & d & e \end{bmatrix} - f \begin{bmatrix} a & b & 0 \\ b & c & d \\ 0 & 0 & f \end{bmatrix}$$

$$= g \begin{bmatrix} a & b & 0 \\ b & c & d \\ 0 & d & e \end{bmatrix} - f^2 \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

Three term recurrence

We can proove it with the *Laplace expansion* (e.g. $n = 4$)

$$T_4 - xS_4 = \begin{pmatrix} a & b & 0 & 0 \\ b & c & d & 0 \\ 0 & d & e & f \\ 0 & 0 & f & g \end{pmatrix}$$

$$det\Big(T_4 - \lambda S_4\Big) = g \begin{bmatrix} a & b & 0 \\ b & c & d \\ 0 & d & e \end{bmatrix} - f \begin{bmatrix} a & b & 0 \\ b & c & d \\ 0 & 0 & f \end{bmatrix}$$

$$= g \begin{bmatrix} a & b & 0 \\ b & c & d \\ 0 & d & e \end{bmatrix} - f^2 \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

Three term recurrence

Oss

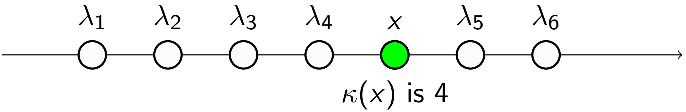
If $x = \lambda$ then $\rho_n(\lambda) = \mathcal{F}_{(T_n, S_n)}(\lambda) = f$

Oss

$\rho_0, \rho_1, \ldots, \rho_n$ i a *Sturm sequence* of polynomials so, $\forall x \in \mathbb{R}$, we have

$$\kappa(x) := \#\text{ eigenvalues less then } x$$

$$\kappa(x) = \#\text{ consecutive sign changes in } \{\rho_i\}_{i=0,\ldots,n}$$



Three term recurrence

Obviusly $f^{'} = \rho_n^{'}, f^{''} = \rho_n^{''}$.

$^{*}\kappa(x) < m$ implies $sign(\lambda_m - x) = +$

Three term recurrence

Obviusly $f' = \rho_n', f'' = \rho_n''$.
So*

$$\text{IF } (\kappa(x_0) \geq m \text{ AND } -\frac{f'}{f} < 0) \text{ OR } (\kappa(x_0) < m \text{ AND } -\frac{f'}{f} \geq 0) \text{ THEN}$$

* $\kappa(x) < m$ implies $sign(\lambda_m - x) = +$

Three term recurrence

Obviusly $f^{'} = \rho_n^{'}, f^{''} = \rho_n^{''}$.
So*

$$\text{IF } (\kappa(x_0) \geq m \text{ AND } -\frac{f^{'}}{f} < 0) \text{ OR } (\kappa(x_0) < m \text{ AND } -\frac{f^{'}}{f} \geq 0) \text{ THEN}$$

We have cubic convergence in Laguerre's method with x_0 as starting point.

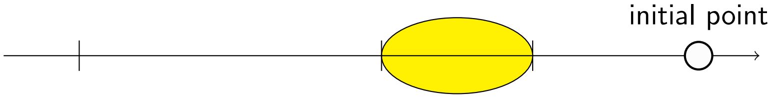
---

*$\kappa(x) < m$ implies $sign(\lambda_m - x) = +$

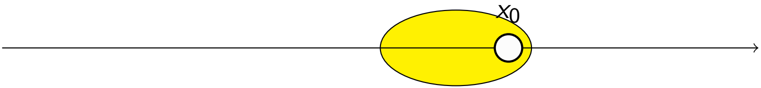
Example

Example

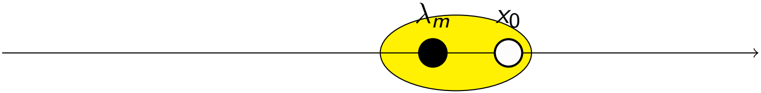
Example



Example



Example



Searching x_0

We want:	Fast and secure iterative method. Starting points for our method. Scalability.
We have:	Laguerre's method. Cuppen's divide and conquer method. Symmetric tridiagonal matrices.
We add:	Unreducible condition. Dynamic programming (Bottom-up). Efficient matrix storing.

Ideas:

If $n = 1$ we have $t \cdot v = \lambda \cdot s \cdot v$. $s \neq 0$, so $\lambda = \frac{t}{s}$.

Ideas:

If $n = 1$ we have $t \cdot v = \lambda \cdot s \cdot v$. $s \neq 0$, so $\lambda = \frac{t}{s}$.

If $n = 2$ we have $\begin{bmatrix} t_{1,1} & t_{1,2} \\ t_{2,1} & t_{2,2} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \lambda \begin{bmatrix} s_{1,1} & s_{1,2} \\ s_{2,1} & s_{2,2} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$

Ideas:

If $n = 1$ we have $t \cdot v = \lambda \cdot s \cdot v$. $s \neq 0$, so $\lambda = \frac{t}{s}$.
If $n = 2$ we have $\begin{bmatrix} t_{1,1} & t_{1,2} \\ t_{2,1} & t_{2,2} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \lambda \begin{bmatrix} s_{1,1} & s_{1,2} \\ s_{2,1} & s_{2,2} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$
calling*

$$S^{-1}T = \delta^{-1} \cdot \begin{pmatrix} \alpha & \gamma \\ \gamma & \beta \end{pmatrix}$$

we resolve $det[S^{-1}T - \lambda I] = 0$ with

$$\lambda_{1,2} = \frac{1}{2\delta}\left(\alpha + \beta \pm \sqrt{\alpha^2 + \beta^2 + \gamma^2 - \alpha\beta}\right)$$

* $\alpha =, \beta =, \gamma =$.

Ideas:

And if $n = 4$?

$$\begin{pmatrix} t_{1,1} & t_{1,2} & 0 & 0 \\ t_{2,1} & t_{2,2} & t_{2,3} & 0 \\ 0 & t_{3,2} & t_{3,3} & t_{3,4} \\ 0 & 0 & t_{4,3} & t_{4,4} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \lambda \begin{pmatrix} s_{1,1} & s_{1,2} & 0 & 0 \\ s_{2,1} & s_{2,2} & s_{2,3} & 0 \\ 0 & s_{3,2} & s_{3,3} & s_{3,4} \\ 0 & 0 & s_{4,3} & s_{4,4} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}$$

Ideas:

And if $n = 4$?

$$\begin{pmatrix} t_{1,1} & t_{1,2} & 0 & 0 \\ t_{2,1} & t_{2,2} & t_{2,3} & 0 \\ 0 & t_{3,2} & t_{3,3} & t_{3,4} \\ 0 & 0 & t_{4,3} & t_{4,4} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \lambda \begin{pmatrix} s_{1,1} & s_{1,2} & 0 & 0 \\ s_{2,1} & s_{2,2} & s_{2,3} & 0 \\ 0 & s_{3,2} & s_{3,3} & s_{3,4} \\ 0 & 0 & s_{4,3} & s_{4,4} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}$$

Ideas:

And if $n = 4$?

$$\begin{pmatrix} t_{1,1} & t_{1,2} & 0 & 0 \\ t_{2,1} & t_{2,2} & 0 & 0 \\ 0 & 0 & t_{3,3} & t_{3,4} \\ 0 & 0 & t_{4,3} & t_{4,4} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \lambda \begin{pmatrix} s_{1,1} & s_{1,2} & 0 & 0 \\ s_{2,1} & s_{2,2} & 0 & 0 \\ 0 & 0 & s_{3,3} & s_{3,4} \\ 0 & 0 & s_{4,3} & s_{4,4} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}$$

Ideas:

And if $n = 4$?

$$\begin{pmatrix} t_{1,1} & t_{1,2} & 0 & 0 \\ t_{2,1} & t_{2,2} & 0 & 0 \\ 0 & 0 & t_{3,3} & t_{3,4} \\ 0 & 0 & t_{4,3} & t_{4,4} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \lambda \begin{pmatrix} s_{1,1} & s_{1,2} & 0 & 0 \\ s_{2,1} & s_{2,2} & 0 & 0 \\ 0 & 0 & s_{3,3} & s_{3,4} \\ 0 & 0 & s_{4,3} & s_{4,4} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}$$

We can solve the blue part and obtain μ_1, μ_2, the cyan part and obtain μ_3, μ_4.

Ideas:

For $j = 1, 2, 3, 4$, from μ_j we can enter in our "best neighborhood" of λ_j using bisection:

Ideas:

For $j = 1, 2, 3, 4$, from μ_j we can enter in our "best neighborhood" of λ_j using bisection:

set $a_j = 0$ and $b_j = \mu_j$

$\#$ Start bisection

set $x = \mu_j$

IF $\kappa(x) < j$

THEN set $a_j = x$

ELSE set $b_j = x$

IF THEN

ELSE go to $\#$

Ideas:

Hopefully we have

$$x_0^1, L(x_0^1), L(L(x_0^1)), \cdots \rightsquigarrow \lambda_1$$

$$x_0^2, L(x_0^2), L(L(x_0^2)), \cdots \rightsquigarrow \lambda_2$$

$$x_0^3, L(x_0^3), L(L(x_0^3)), \cdots \rightsquigarrow \lambda_3$$

$$x_0^4, L(x_0^4), L(L(x_0^4)), \cdots \rightsquigarrow \lambda_4$$

Split

Consider $(\hat{T}, \hat{S})$ with

$$\hat{T} = \begin{bmatrix} T_0 & 0 \\ 0 & T_1 \end{bmatrix}$$
$$\hat{S} = \begin{bmatrix} S_0 & 0 \\ 0 & S_1 \end{bmatrix}$$

and let be

$$\hat{\lambda}_1 \leq \hat{\lambda}_2 \leq \cdots \leq \hat{\lambda}_n$$

eigenvalues of $(\hat{T}, \hat{S})$, then

Split

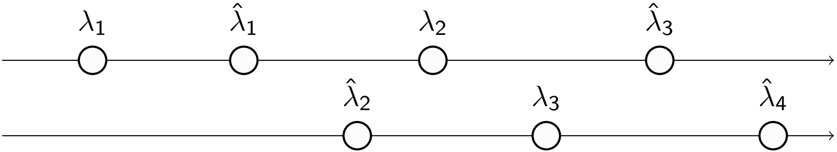
Teo (A sort of "interlacing")

$$
\begin{aligned}
-\infty <&\lambda_1 \le \hat{\lambda}_1 \\
\hat{\lambda}_{i-1} \le&\lambda_i \le \hat{\lambda}_{i+1} \\
\hat{\lambda}_n \le&\lambda_n < \infty
\end{aligned}
$$

with $i = 2, 3, \ldots, n-1$.

Oss

It's possible that



Split

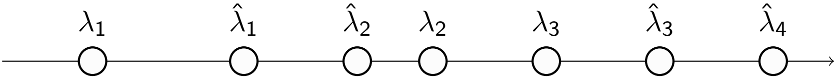
Teo (A sort of "interlacing")

$$
\begin{aligned}
-\infty <&\lambda_1 \le \hat{\lambda}_1 \\
\hat{\lambda}_{i-1} \le&\lambda_i \le \hat{\lambda}_{i+1} \\
\hat{\lambda}_n \le&\lambda_n < \infty
\end{aligned}
$$

with $i = 2, 3, \ldots, n-1$.

Oss

It's possible that



Split

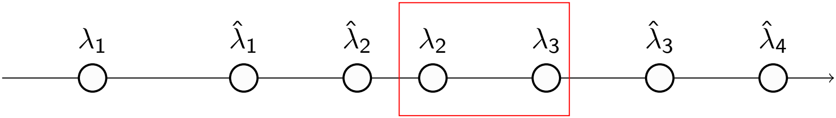
Teo (A sort of “interlacing”)

$$\begin{aligned} -\infty < & \lambda_1 \le \hat{\lambda}_1 \\ \hat{\lambda}_{i-1} \le & \lambda_i \le \hat{\lambda}_{i+1} \\ \hat{\lambda}_n \le & \lambda_n < \infty \end{aligned}$$

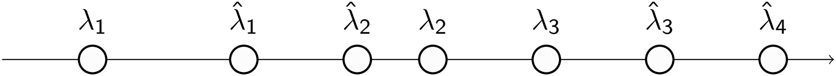
with $i = 2, 3, \ldots, n-1$.

Oss

It’s possible that



Split



So we can't

Grazie per l'attenzione.