

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 26, 2013

Generalized Eigenvalue Problem

Def

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

We consider *only*

symmetric and tridiagonal

pencil.

Generalized Eigenvalue Problem

Def

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

We consider *only* pencil.

symmetric
and
tridiagonal

S is *positive definite*.

Generalized Eigenvalue Problem

Def

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

We consider *only* pencil.

symmetric
and
tridiagonal

S is *positive definite*.

Def (Problem)

Find $\lambda \in \mathbb{R}$ such that $Tv = \lambda Sv$, 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 [?])

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

Unreducible pencil

Def (as in [?])

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

exemplum gratie:

Bad

$$\begin{aligned} T &= I, S = 0 \\ T &= I, S = I \end{aligned}$$

Good

$$\begin{aligned} T &= I, S = \text{trid}(-1, 2, -1) \\ T &= \text{trid}(-1, 2, -1), S = \text{trid}(-1, 2, -1) \\ T &= \text{trid}(\text{rnd}_{\text{sub}}, \text{rnd}_{\text{diag}}, \text{rnd}_{\text{sub}}), S = I, \\ &\text{with } \text{rnd}_{\text{sub}} \text{ random number } \neq 0. \end{aligned}$$

Matrix storin

$$T = \text{trid}(\text{sub}, \text{diag}, \text{super})$$

But T is symmetric, so $\text{sub} = \text{super}$. We define and use

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

Listing 1: T, S as couple of array

with $T(:,0) = \text{diag}$ and $T(:,1) = \text{super}$.

Remark

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.

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 x between them and the quadric

Fast and secure iterative method

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

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$$

where we count with multiplicity.

If λ_m and λ_{m+1} are simple zeros (mlt=1), then we consider x between them and the quadric

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

Fast and secure iterative method

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

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$$

where we count with multiplicity.

If λ_m and λ_{m+1} are simple zeros (mlt=1), then we consider x between them and 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 $g_u(\lambda_m), g_u(\lambda_{m+1}) > 0$.

So we have two sign change.

Fast and secure iterative method

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

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$$

where we count with multiplicity.

If λ_m and λ_{m+1} are simple zeros (mlt=1), then we consider x between them and 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 $g_u(\lambda_m), g_u(\lambda_{m+1}) > 0$.

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:

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

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

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

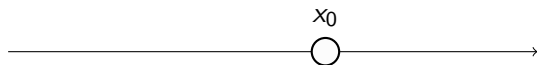
So we need $\frac{f'}{f}$ and $\frac{f''}{f}$.

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**.

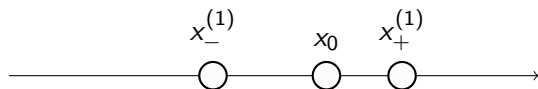
Fast and secure iterative method

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

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

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

else we have a similar expression.

Fast and secure iterative method

Def (Laguerre's iteration)

If $m/t(\lambda_m) = m/t(\lambda_{m+1}) = 1$ 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.

► proof We can prove that

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

Fast and secure iterative method

Def (Laguerre's iteration)

If $mIt(\lambda_m) = mIt(\lambda_{m+1}) = 1$ 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.

► proof We can prove that

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

So Laguerre's method is **secure**.

Fast and secure iterative method

We also have an important property:

Teo

If we choose $\lambda_m < x_0$ s.t. $\text{sign}\left(\frac{f'(x_0)}{f(x_0)}\right) = \text{sign}(\lambda_m - x_0)$ then*

$\{x_-^{(k)}\}_{k=1,\dots}$ converge monotonically in asymptotically cubic rate to λ_m .

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

Fast and secure iterative method

We also have an important property:

Teo

If we choose* $\lambda_m < x_0$ s.t. $\text{sign}\left(\frac{f'(x_0)}{f(x_0)}\right) = \text{sign}(\lambda_m - x_0)$ then

$\{x_-^{(k)}\}_{k=1,\dots}$ converge monotonically in asymptotically cubic rate to λ_m .

So we can exactly define a neighborhood “near” λ and in it we have cubic rate convergence (much, much faster than **simple bisection**)

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

Fast and secure iterative method

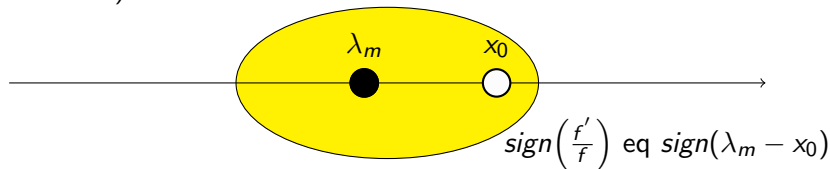
We also have an important property:

Teo

If we choose* $\lambda_m < x_0$ s.t. $\text{sign}\left(\frac{f'(x_0)}{f(x_0)}\right) = \text{sign}(\lambda_m - x_0)$ then

$\{x_{-}^{(k)}\}_{k=1,\dots}$ converge monotonically in asymptotically cubic rate to λ_m .

So we can exactly define a neighborhood “near” λ and in it we have cubic rate convergence (much, much faster than **simple bisection**)



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

Fast and secure iterative method

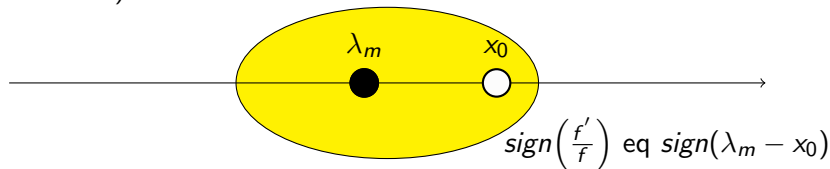
We also have an important property:

Teo

If we choose* $\lambda_m < x_0$ s.t. $\text{sign}\left(\frac{f'(x_0)}{f(x_0)}\right) = \text{sign}(\lambda_m - x_0)$ then

$\{x_{-}^{(k)}\}_{k=1,\dots}$ converge monotonically in asymptotically cubic rate to λ_m .

So we can exactly define a neighborhood “near” λ and in it we have cubic rate convergence (much, much faster than **simple bisection**)



The Laguerre's method is **fast**.

*for $x_0 < \lambda_{m+1}$ s.t. $\text{sign}\left(\frac{f'(x_0)}{f(x_0)}\right) = \text{sign}(\lambda_m - x_0)$ 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 powerful method to obtain x_0 and an algorithm to estimate $mlt(\lambda_m)$.

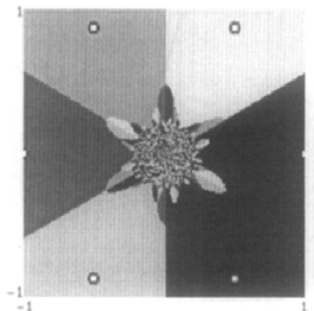
Overestimate $mlt(\lambda_m)$ (as we can read in [?]) causes no trouble, so the most important aspects of our calculation are:

Good x_0 .

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

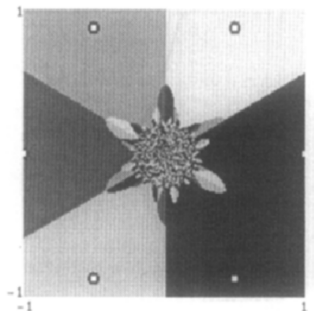
Complex case (little digression)

According to [?], we observe that searching $z \in \mathbb{C}$ such that $z^n - 1 = 0$ for $n > 4$ with Laguerre's method it's difficult because near the origin there is a Julia fractal set for starting point z_0 .
(figure: $n = 6$)



Complex case (little digression)

According to [?], we observe that searching $z \in \mathbb{C}$ such that $z^n - 1 = 0$ for $n > 4$ with Laguerre's method it's difficult because near the origin there is a Julia fractal set for starting point z_0 .
(figure: $n = 6$)



So if we want to solve the Generalized Eigenvalues Problem with $T, S \in \mathbb{C}$ we have **great problems** to place the starting point if $n > 4$.

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

$$\rho_0 := 1, \rho_1 := t_{1,1} - x s_{1,1}$$

$$\rho_i := (t_{i,i} - x s_{i,i}) \rho_{i-1} - (t_{i-1,i} - x s_{i-1,i})^2 \rho_{i-2}, \quad i = 2, 3, \dots, n$$

Three term recurrence

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

$$T_4 - \lambda S_4 = \begin{pmatrix} a & b & 0 & 0 \\ b & c & d & 0 \\ 0 & d & e & f \\ 0 & 0 & f & g \end{pmatrix}$$

$$\begin{aligned} \det(T_4 - \lambda S_4) &= g \begin{vmatrix} a & b & 0 \\ b & c & d \\ 0 & d & e \end{vmatrix} - f \begin{vmatrix} a & b & 0 \\ b & c & d \\ 0 & 0 & f \end{vmatrix} \\ &= g \begin{vmatrix} a & b & 0 \\ b & c & d \\ 0 & d & e \end{vmatrix} - f^2 \begin{vmatrix} a & b \\ b & c \end{vmatrix} \end{aligned}$$

Three term recurrence

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

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

$$\begin{aligned} \det(T_4 - \lambda S_4) &= g \begin{bmatrix} a & b & 0 \\ b & c & d \\ 0 & d & e \end{bmatrix} - f \begin{bmatrix} a & b & 0 \\ b & c & d \\ 0 & 0 & f \end{bmatrix} \\ &= g \begin{bmatrix} a & b & 0 \\ b & c & d \\ 0 & d & e \end{bmatrix} - f^2 \begin{bmatrix} a & b \\ b & c \end{bmatrix} \end{aligned}$$

Three term recurrence

Remark

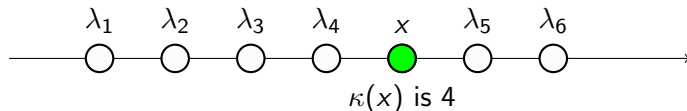
If $x = \lambda$ then $\rho_n(\lambda) = \mathcal{F}_{(T_n, S_n)}(\lambda) = f(\lambda)$

Remark (important results with [▶ proof](#) in Appendice)

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

$\kappa(x) := \#$ eigenvalues less than x

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



Three term recurrence

Obviously $f' = \rho'_n, f'' = \rho''_n$.

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

Three term recurrence

Obviously $f' = \rho'_n, f'' = \rho''_n$.

So*

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

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

Three term recurrence

Obviously $f' = \rho'_n, f'' = \rho''_n$.

So*

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

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

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

Example

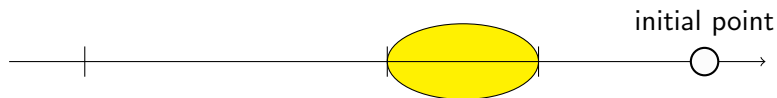
initial point



Example



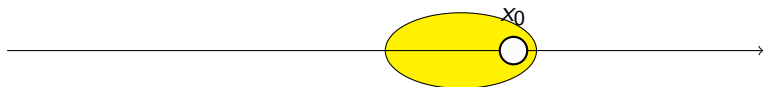
Example



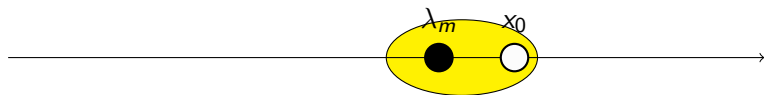
with **bisection** we can find the neighbourhood of λ_m

Example

we can now use the Laguerre's iteration



Example



What we really calculate is...

We **define** $\xi_i = \frac{\rho_i}{\rho_{i-1}}$ for $i = 2, \dots, n$ and we have $\rho_i = \prod_{k=1}^n \xi_k$.

What we really calculate is...

We **define** $\xi_i = \frac{\rho_i}{\rho_{i-1}}$ for $i = 2, \dots, n$ and we have $\rho_i = \prod_{k=1}^n \xi_k$.

We also **define** $\eta_i = \frac{\rho_i'}{\rho_i}$, $\zeta_i = \frac{\rho_i''}{\rho_i}$ for $i = 0, 1, \dots, n$ and we finally we have


$\kappa(x) =$ number of **negative terms** in $\{\xi_i\}_{i=1}^n$

$$-\frac{f'(x)}{f(x)} = \eta_n$$

$$\frac{f''(x)}{f(x)} = \zeta_n$$

What we really calculate is...


So we have to calculate three three-term-recurrences.

*we consider all elements of diag and super of (T, S) as non zero. 

What we really calculate is...

So we have to calculate three three-term-recurrences.

* Total $\leq 2 + 38n$ multiplications.


* we consider all elements of diag and super of (T, S) as non zero. 

What we really calculate is...

So we have to calculate three three-term-recurrences.

* Total $\leq 2 + 38n$ multiplications.

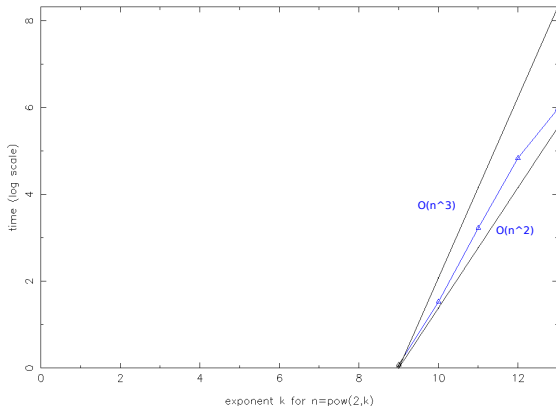
For every step in Laguerre's iteration we have to do $2 + 7 + 38n$ (at most) multiplications and 1 square root extraction. Because the convergence is cubic we **hope** in a small number of iteration.

* we consider all elements of diag and super of (T, S) as non zero. 

Complexity

* Author's prevision: $O(n^2)$.

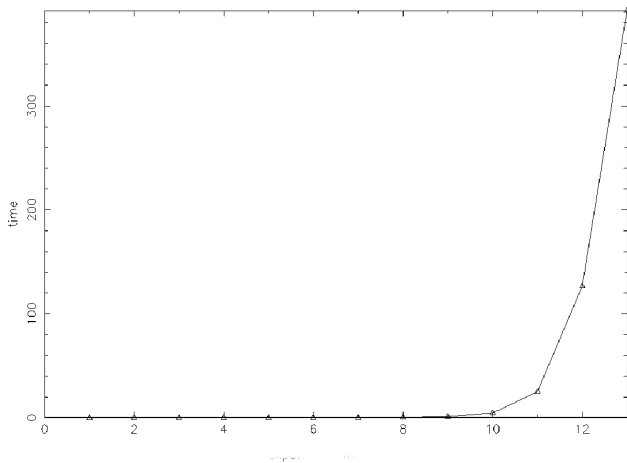
With (I, S) and eigenvalues between 0.5 and 1.5 we have



* They use $n < 100$,
my experiment ends up with $n = 2^{13} = 8192$.

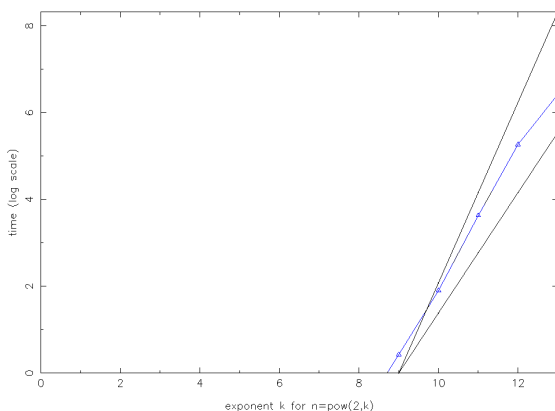
Complexity

And in linear scale:



Complexity

With $T = (a_i, 1, a_i)$, $S = (b_i, c_i, b_i)$ and a_i, b_i random numbers such that $|a_i| \leq 10^{-3}$, $|b_i| \leq 10^{-1}$, $c_i = i10^{-1} + |2b_i|$ we have



We can appreciate little differences between $T = (0, 1, 0)$ and $T = (a_i, 1, a_i)$

Error Analysis

It's a common knowledge that three-term-recurrences suffer from severe overflow and underflow problems, and because of that we use ξ_i instead of ρ_i :

Error Analysis

It's a common knowledge that three-term-recurrences suffer from severe overflow and underflow problems, and because of that we use ξ_i instead of ρ_i :

$\xi_i = \frac{\rho_i}{\rho_{i-1}}$, it's *self-scaled*.

Error Analysis

It's a common knowledge that three-term-recurrences suffer from severe overflow and underflow problems, and because of that we use ξ_i instead of ρ_i :

$\xi_i = \frac{\rho_i}{\rho_{i-1}}$, it's *self-scaled*.

Teo

$$fl[f(x)] = fl[\det[T - xS]] = (1 + \gamma)\det[(T + \delta T) - x(S + \delta S)]$$

where $|\gamma| \leq n\epsilon$, with ϵ machine precision, and both δT and δS are symmetric tridiagonal matrices satisfying entrywise inequalities $|\delta T|_\infty \leq 2.51\epsilon|T|_\infty + \sqrt{\epsilon_u}$, $|\delta S|_\infty \leq 3.51\epsilon|S|_\infty$, where ϵ_u is the underflow threshold (in double precision is 10^{-308}).

Error Analysis

We are interested not only in $\text{fl}[f(x)] = \text{fl}\left[\prod_{i=1}^n \text{fl}[\xi_i]\right]$, but also in $\text{fl}[\eta_n]$ and $\text{fl}[\zeta_n]$, because we use these three value to calculate the Laguerre's iteration.

Error Analysis

We are interested not only in $\text{fl}[f(x)] = \text{fl}\left[\prod_{i=1}^n \text{fl}[\xi_i]\right]$, but also in $\text{fl}[\eta_n]$ and $\text{fl}[\zeta_n]$, because we use these three value to calculate the Laguerre's iteration.

The authors doesn't report this important aspect of analysis.

Error Analysis

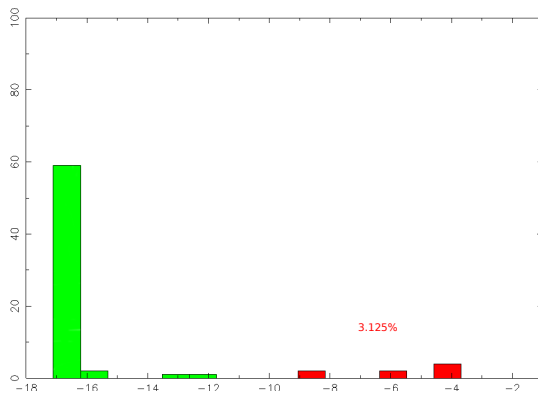
We are interested not only in $\text{fl}[f(x)] = \text{fl}\left[\prod_{i=1}^n \text{fl}[\xi_i]\right]$, but also in $\text{fl}[\eta_n]$ and $\text{fl}[\zeta_n]$, because we use these three value to calculate the Laguerre's iteration.

The authors doesn't report this important aspect of analysis.

($\text{fl}[\eta_n]$, $\text{fl}[\zeta_n]$ suffer from similar problems and have similar backward analysis, but they are -in partucular ζ_n - enourmous in magnitude, sometimes this can cause troubles)

Exponent of absolute error ($n = 256$)

With (I, S) and eigenvalue between 0.5 and 1.5 we have



Where is the problem?

Sometimes (about 3 over 100 eigenvalues with $n \geq 256$)
Laguerre's iteration **break the monotonic convergence!**

Where is the problem?

Sometimes (about 3 over 100 eigenvalues with $n \geq 256$)
Laguerre's iteration **break the monotonic convergence!**
In these cases we have slower convergence and greater error.

Where is the problem?

Sometimes (about 3 over 100 eigenvalues with $n \geq 256$)

Laguerre's iteration **break the monotonic convergence!**

In these cases we have slower convergence and greater error.

I think the problem is in κ, η_n, ζ_n .

Sometimes, when $|\lambda_j - x_0| \leq 10^{-4}$ and (T, S) have small elements, we can have troubles.

Where is the problem?

Sometimes (about 3 over 100 eigenvalues with $n \geq 256$)

Laguerre's iteration **break the monotonic convergence!**

In these cases we have slower convergence and greater error.

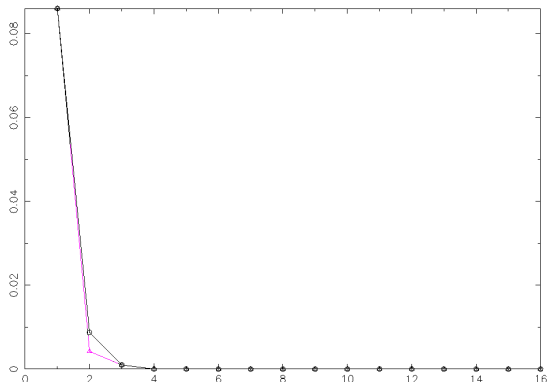
I think the problem is in κ, η_n, ζ_n .

Sometimes, when $|\lambda_j - x_0| \leq 10^{-4}$ and (T, S) have small elements, we can have troubles.

If we compare $\kappa(x_i)$ and the same number **obtained with inertia** for $|\lambda_j - x_i| \leq 10^{-i}$ and $i = 1, \dots, 16$ we have

Where is the problem?

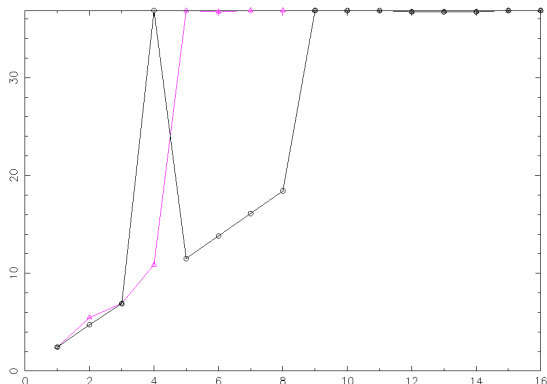
$n = 256$, $\dim = 64$ and $\text{pencil} = (T(193 : 4\dim), S(193 : 4\dim))$



For $i = 1, \dots, 16$, absolute errors between λ_{32} and x_i with κ ,
absolute errors between λ_{32} and x_i with *inertia*.

Where is the problem?

$n = 256$, $\dim = 64$ and $\text{pencil} = (T(193 : 4\dim), S(193 : 4\dim))$



For $i = 1, \dots, 16$, absolute errors between λ_{32} and x_i with κ ,
absolute errors between λ_{32} and x_i with *inertia*.

I use a different scale: $\log\left(\frac{1}{|\text{error}|}\right)$

Where is the problem?

When x_0 for Laguerre's iteration is into our **good neighbourhood** but $10^{-9} \leq |x_0 - \lambda_m| \leq 10^{-1}$, we can have troubles...

Where is the problem?

When x_0 for Laguerre's iteration is into our **good neighbourhood** but $10^{-9} \leq |x_0 - \lambda_m| \leq 10^{-1}$, we can have troubles...
Starting points are the most important aspect of our calculation.

Searching initial points

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 $t \cdot v = \lambda \cdot s \cdot v$ with $s \neq 0$, so $\lambda = \frac{t}{s}$.

Ideas:

If $n = 1$ we have $t \cdot v = \lambda \cdot s \cdot v$ with $s \neq 0$, 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 $t \cdot v = \lambda \cdot s \cdot v$ with $s \neq 0$, 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

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

* $\alpha = s_{2,2}t_{1,1} - s_{1,2}t_{1,2}$, $\beta = -s_{1,2}t_{1,2} + s_{1,1}t_{2,2}$
 $\gamma = -s_{1,2}t_{1,2} + s_{1,1}t_{1,2}$, $\delta = s_{1,1}s_{2,2} - s_{1,2}^2$.

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:

From μ_j we can enter in our neighbourhood of λ_j using simple [bisection](#):

*This is an important part of our method.

Authors of [?, ?] doesn't explain this point.

► code

Ideas:

From μ_j we can enter in our neighbourhood of λ_j using simple bisection:

Pseudocode:

```
set  $a_j = 0$  and  $b_j = \mu_j$ 
set  $x = \mu_j$ 
# IF  $\kappa(x) < j$ 
THEN set  $a_j = x$ 
ELSE set  $b_j = x$ 
*IF  $-\frac{f'(x)}{f(x)} = \text{sign}(\lambda_j - x)$ 
THEN stop
ELSE set  $x = \frac{b_j - a_j}{2}$  and go to #
set  $x_0 = x$ 
```

*This is an important part of our method.

Authors of [?, ?] doesn't explain this point.

[code](#)

Ideas:

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

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

Hopefully we have

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

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

Split

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

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

with T_0, T_1, S_0, S_1 symmetric tridiagonal, and let be

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

eigenvalues of (\hat{T}, \hat{S}) , they are what we call μ_1, \dots, μ_n .

Split

Teo (A sort of “interlacing” with [proof](#))

$$-\infty < \lambda_1 \leq \hat{\lambda}_1$$

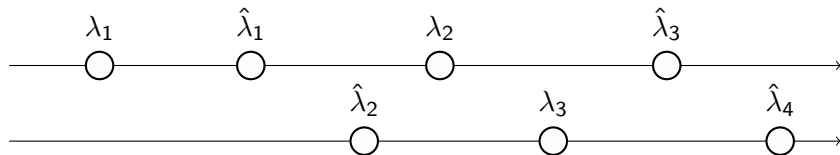
$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

$$\hat{\lambda}_n \leq \lambda_n < \infty$$

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

Remark

It's possible that



Split

Teo (A sort of “interlacing” with [▶ proof](#))

$$-\infty < \lambda_1 \leq \hat{\lambda}_1$$

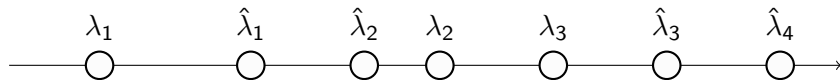
$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

$$\hat{\lambda}_n \leq \lambda_n < \infty$$

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

Remark

It's possible that



Split

Teo (A sort of “interlacing” with [proof](#))

$$-\infty < \lambda_1 \leq \hat{\lambda}_1$$

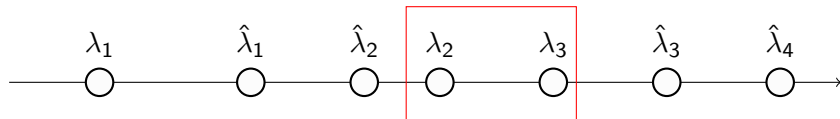
$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

$$\hat{\lambda}_n \leq \lambda_n < \infty$$

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

Remark

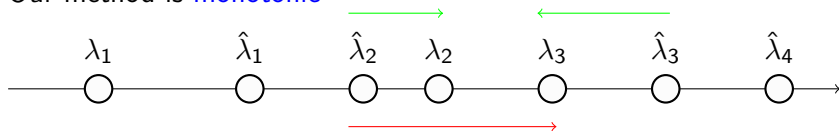
It's possible that



Split

Remark

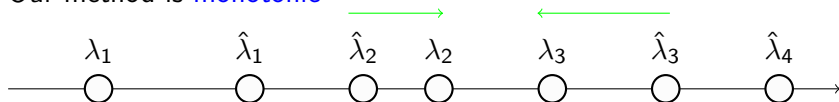
Our method is **monotonic**



Split

Remark

Our method is **monotonic**

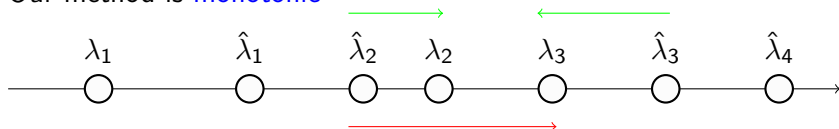


We can reach λ_2 only from $\hat{\lambda}_2$, moving from left to right (and similar λ_3 only from $\hat{\lambda}_3$, moving from right to left).

Split

Remark

Our method is **monotonic**



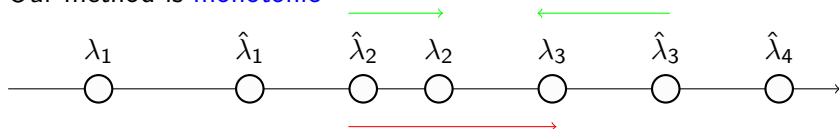
We can reach λ_2 only from $\hat{\lambda}_2$, moving from left to right (and similar λ_3 only from $\hat{\lambda}_3$, moving from right to left).

If $|\lambda_2 - \lambda_3| \leq 10^{-14}$ we can have troubles.

Split

Remark

Our method is **monotonic**



We can reach λ_2 only from $\hat{\lambda}_2$, moving from left to right (and similar λ_3 only from $\hat{\lambda}_3$, moving from right to left).

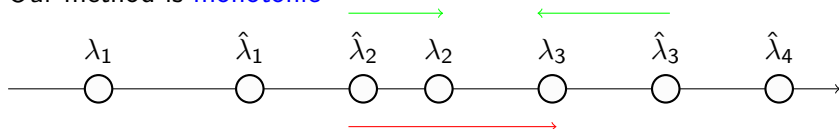
If $|\lambda_2 - \lambda_3| \leq 10^{-14}$ we can have troubles.

Luckily we also have a *multiplicity estimator* (called EstMlt) that works only with $\hat{\lambda}_1, \dots, \hat{\lambda}_n$.

Split

Remark

Our method is **monotonic**



We can reach λ_2 only from $\hat{\lambda}_2$, moving from left to right (and similar λ_3 only from $\hat{\lambda}_3$, moving from right to left).

If $|\lambda_2 - \lambda_3| \leq 10^{-14}$ we can have troubles.

Luckily we also have a *multiplicity estimator* (called EstMlt) that works only with $\hat{\lambda}_1, \dots, \hat{\lambda}_n$.

If $m/t = 2$ then we consider $\lambda_2 = \lambda_3$, i.e. we said that λ_2 has multiplicity 2.

EstMlt

We have $j, x, \operatorname{sgn}\left(-\frac{f'(x)}{f(x)}\right)$ and $\hat{\lambda}_1, \dots, \hat{\lambda}_n$ as *INPUT*.

Pseudocode:

```
mlt = 1
do  $k = 1, \dots$ 
   $m = j + k \operatorname{sgn}\left(-\frac{f'(x)}{f(x)}\right)$ 
  if  $m \leq 0$  then
    something goes wrong
  set  $mlt = 1$ 
  go to #
end if
if  $|\hat{\lambda}_j - \hat{\lambda}_m| \leq 0.01|\hat{\lambda}_j - x|$  then
   $mlt = mlt + 1$ 
else go to #
end if
end do
# stop
```

Dynamic programming

We want:

Fast and secure iterative method.
Starting points for our method.
Scalability.

We have:

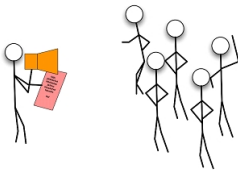
Laguerre's method.
Cuppen's divide and conquer method.
Symmetric tridiagonal matrices.

We add:

Unreducible condition.
Dynamic programming (Bottom-up).
Efficient matrix storing.

Dynamic programming

The author's point of view is **parallel computing**.



Dynamic programming

My point of view is **sequential computing**.



Dynamic programming

Instead of [recursion](#) I chose [dynamic programming](#), *i.e.*

“solving complex problems by breaking them down into simpler subproblems. [...] In general, to solve a given problem, we need to solve different parts of the problem (subproblems), then combine the solutions of the subproblems to reach an overall solution.”

(cit. Wikipedia)

Dynamic programming

Instead of [recursion](#) I chose [dynamic programming](#), *i.e.*

“solving complex problems by breaking them down into simpler subproblems. [...] In general, to solve a given problem, we need to solve different parts of the problem (subproblems), then combine the solutions of the subproblems to reach an overall solution.”

(cit. Wikipedia)

If $n = 2^k$, to compute eigenvalues of (T, S) we use (\hat{T}, \hat{S}) , that use $(\hat{\hat{T}}, \hat{\hat{S}})$, that use $(\hat{\hat{\hat{T}}}, \hat{\hat{\hat{S}}})$, ...

Thank you.

Appendice A: proof of Laguerre's convergence

According to [?, p.444] we can write

$$x_{\pm}^{(k+1)} = x_{\pm}^{(k)} - \frac{nf}{f' \pm H^{\frac{1}{2}}}$$
$$H = (n-1)^2(f')^2 - n(n-1)ff''$$

If we choose the sign so that the $|f' \pm H^{\frac{1}{2}}|$ has the larger absolute value then we can approx $x_{\pm}^{(k+1)} - \lambda_m$ as

$$x_{\pm}^{(k+1)} - \lambda_m \approx \frac{1}{2}(x_{\pm}^{(k)} - \lambda_m)^3 \frac{(n-1)\Sigma'_2 - (\Sigma'_1)^2}{n-1}$$
$$\Sigma'_2 = \sum_{i \neq n} \frac{1}{(\lambda_m - \lambda_i)^2}$$
$$\Sigma'_1 = \sum_{i \neq n} \frac{1}{\lambda_m - \lambda_i}$$

Appendice A: proof of Laguerre's convergence

So we have **convergence** and $x_{\pm}^{(k+1)} - \lambda_m \approx \text{number } (x_{\pm}^{(k)} - \lambda_m)^3$
tell us that the convergence is **cubic**.

► back

Appendice B: proof of property about Sturm sequence

► back

Appendice B: interlacing

Def

For $\alpha \in [0, 1]$ we define the pencil

$$(T(\alpha), S(\alpha)) := ((1 - \alpha)\hat{T} + \alpha T, (1 - \alpha)\hat{S} + \alpha S).$$

Lemm

$(T(\alpha), S(\alpha))$ is a symmetric definite pencil for each $\alpha \in [0, 1]$.

Calling $\lambda_1(\alpha) \leq \lambda_2(\alpha) \leq \dots \leq \lambda_n(\alpha)$ the n real eigenvalues of the pencil $(T(\alpha), S(\alpha))$ we have

Lemm

Each $\lambda_i(\alpha)$ is a continuous function of $\alpha \in [0, 1]$.

Appendice B: interlacing

Teo (A sort of “interlacing”)

$$-\infty < \lambda_1 \leq \hat{\lambda}_1$$

$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

$$\hat{\lambda}_n \leq \lambda_n < \infty$$

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

Appendice B: interlacing

Teo (A sort of “interlacing”)

$$-\infty < \lambda_1 \leq \hat{\lambda}_1$$

$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

$$\hat{\lambda}_n \leq \lambda_n < \infty$$

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

Classic interlacing still works for $-\infty < \lambda_1 \leq \hat{\lambda}_1$ and for $\hat{\lambda}_n \leq \lambda_n < \infty$.

Appendice B: interlacing

Teo (A sort of “interlacing”)

$$-\infty < \lambda_1 \leq \hat{\lambda}_1$$

$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

$$\hat{\lambda}_n \leq \lambda_n < \infty$$

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

Classic interlacing still works for $-\infty < \lambda_1 \leq \hat{\lambda}_1$ and for $\hat{\lambda}_n \leq \lambda_n < \infty$.

We have to prove that $\lambda_i \geq \hat{\lambda}_{i-1}$ (and similar $\lambda_i \leq \hat{\lambda}_{i+1}$).

Appendice B: interlacing

(Proof by contradiction) if we consider $\lambda_i < \hat{\lambda}_{i-1}$ for some $i \in \{2, 3, \dots, n-1\}$ (that, in our new writing, is $\lambda_i(1) < \lambda_{i-1}(0)$) because all $\lambda_j(1)$ are eigenvalues of (T, S) and all $\lambda_j(0)$ are eigenvalues of (\hat{T}, \hat{S}) then

Appendix B: interlacing

(Proof by contradiction) if we consider $\lambda_i < \hat{\lambda}_{i-1}$ for some $i \in \{2, 3, \dots, n-1\}$ (that, in our new writing, is $\lambda_i(1) < \lambda_{i-1}(0)$) because all $\lambda_j(1)$ are eigenvalues of (T, S) and all $\lambda_j(0)$ are eigenvalues of (\hat{T}, \hat{S}) then



Appendice B: interlacing

(Proof by contradiction) if we consider $\lambda_i < \hat{\lambda}_{i-1}$ for some $i \in \{2, 3, \dots, n-1\}$ (that, in our new writing, is $\lambda_i(1) < \lambda_{i-1}(0)$) because all $\lambda_j(1)$ are eigenvalues of (T, S) and all $\lambda_j(0)$ are eigenvalues of (\hat{T}, \hat{S}) then



in symbols: $\lambda_{i-1}(1) \leq \lambda_i(1) < \lambda_{i-1}(0) \leq \lambda_i(0)$.

Appendix B: interlacing

(Proof by contradiction) if we consider $\lambda_i < \hat{\lambda}_{i-1}$ for some $i \in \{2, 3, \dots, n-1\}$ (that, in our new writing, is $\lambda_i(1) < \lambda_{i-1}(0)$) because all $\lambda_j(1)$ are eigenvalues of (T, S) and all $\lambda_j(0)$ are eigenvalues of (\hat{T}, \hat{S}) then



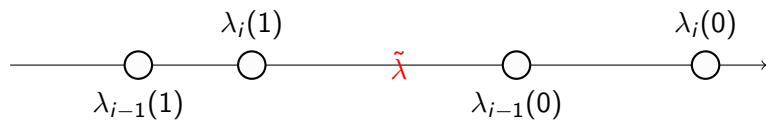
in symbols: $\lambda_{i-1}(1) \leq \lambda_i(1) < \lambda_{i-1}(0) \leq \lambda_i(0)$.

For all $\tilde{\lambda} \in [\lambda_i(1), \lambda_{i-1}(0)]$ we can find α_i, α_{i-1} such as $\tilde{\lambda} = \lambda_i(\alpha_i) = \lambda_{i-1}(\alpha_{i-1})$.

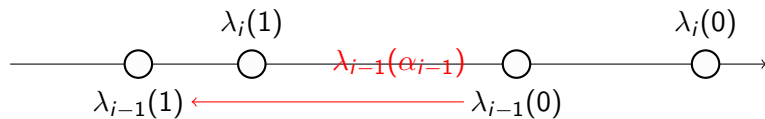
Appendice B: interlacing



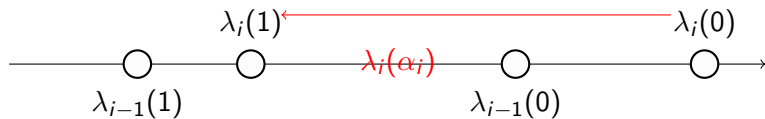
Appendice B: interlacing



Appendice B: interlacing



Appendice B: interlacing



Appendice B: interlacing

$$H(\alpha, \lambda) := \det[T(\alpha) - \lambda S(\alpha)] =$$

$$\begin{bmatrix} \ddots & & \ddots & & \\ \ddots & t_{k,k} - \lambda s_{k,k} & \alpha(t_{k,k+1} - s_{k,k+1}) & & \\ \alpha(t_{k,k+1} - s_{k,k+1}) & t_{k+1,k+1} - \lambda s_{k+1,k+1} & & \ddots & \\ & & \ddots & & \ddots \end{bmatrix}$$

Appendice B: interlacing

$$H(\alpha, \lambda) := \det[T(\alpha) - \lambda S(\alpha)] =$$

$$\begin{bmatrix} \ddots & & \ddots & & \\ \ddots & t_{k,k} - \lambda s_{k,k} & \alpha(t_{k,k+1} - s_{k,k+1}) & & \\ \alpha(t_{k,k+1} - s_{k,k+1}) & t_{k+1,k+1} - \lambda s_{k+1,k+1} & & \ddots & \\ & & \ddots & \ddots & \end{bmatrix}$$

So

$$H(\alpha, \lambda) = p(\lambda) + \alpha^2 q(\lambda)$$

with p, q polynomials.

Appendice B: interlacing

If $q(\tilde{\lambda}) \neq 0$ then $\tilde{\alpha} = +\sqrt{H(\tilde{\alpha}, \tilde{\lambda}) - \frac{p(\tilde{\lambda})}{q(\tilde{\lambda})}} \in [0, 1]$.
only **one** solution in α .

Appendice B: interlacing

If $q(\tilde{\lambda}) \neq 0$ then $\tilde{\alpha} = +\sqrt{H(\tilde{\alpha}, \tilde{\lambda}) - \frac{p(\tilde{\lambda})}{q(\tilde{\lambda})}} \in [0, 1]$.

only **one** solution in α .

But we have, **for all** $\tilde{\lambda}$, **two** solution in α , named α_j and α_{j-1} .

Appendice B: interlacing

If $q(\tilde{\lambda}) \neq 0$ then $\tilde{\alpha} = +\sqrt{H(\tilde{\alpha}, \tilde{\lambda}) - \frac{p(\tilde{\lambda})}{q(\tilde{\lambda})}} \in [0, 1]$.

only **one** solution in α .

But we have, **for all** $\tilde{\lambda}$, **two** solution in α , named α_j and α_{j-1} .

Then $q(\lambda) = 0$ for all $\tilde{\lambda}$.

We have $\tilde{\lambda}$ eigenvalue of (\hat{T}, \hat{S}) , **for all** $\tilde{\lambda}$ and (\hat{T}, \hat{S}) have exactly n eigenvalues.

Appendice B: interlacing

If $q(\tilde{\lambda}) \neq 0$ then $\tilde{\alpha} = +\sqrt{H(\tilde{\alpha}, \tilde{\lambda}) - \frac{p(\tilde{\lambda})}{q(\tilde{\lambda})}} \in [0, 1]$.

only **one** solution in α .

But we have, **for all** $\tilde{\lambda}$, **two** solution in α , named α_i and α_{i-1} .

Then $q(\lambda) = 0$ for all $\tilde{\lambda}$.

We have $\tilde{\lambda}$ eigenvalue of (\hat{T}, \hat{S}) , **for all** $\tilde{\lambda}$ and (\hat{T}, \hat{S}) have exactly n eigenvalues.

contradiction

► back

Appendice D: code

► back

Grazie (ancora) per l'attenzione.