

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 16, 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

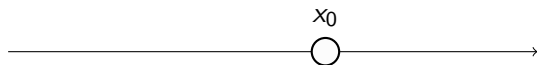
So we need $\frac{f'}{f}$ and $\frac{f''}{f}$.

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**.

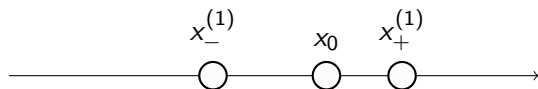
Fast and secure iterative method

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



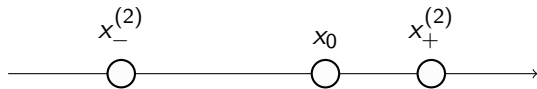
Fast and secure iterative method

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

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

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

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

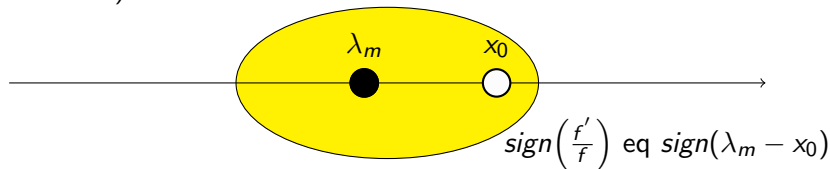
We also have an important property:

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

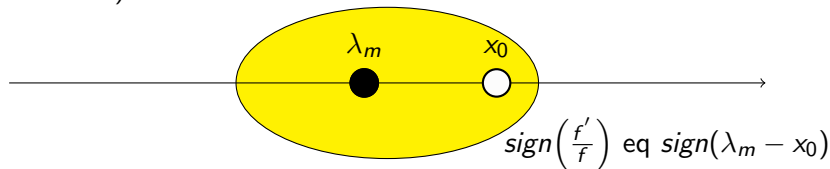
We also have an important property:

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 $mlt(\lambda_m)$.

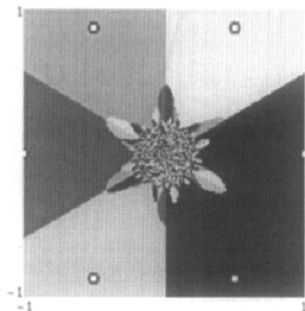
Overestimate $mlt(\lambda_m)$ (as we can read in [?]) causes no trouble, so the most important aspects of our calculation are:

Good x_0 .

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

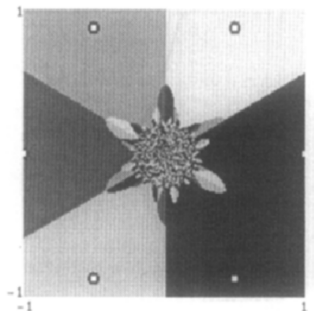
Complex case (little digression)

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. $n = 4$)

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

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

Three term recurrence

Remark

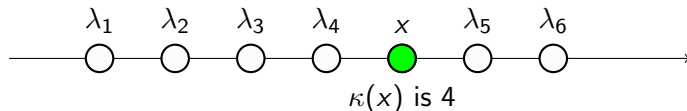
If $x = \lambda$ then $\rho_n(\lambda) = \mathcal{F}_{(T_n, S_n)}(\lambda) = f(\lambda)$

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

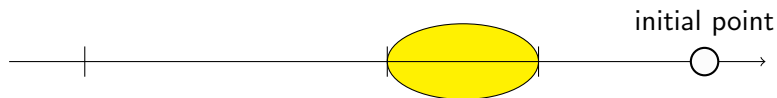
initial point



Example



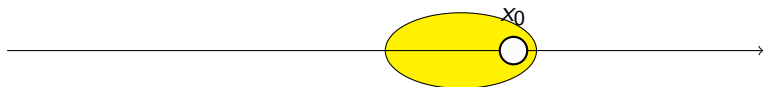
Example



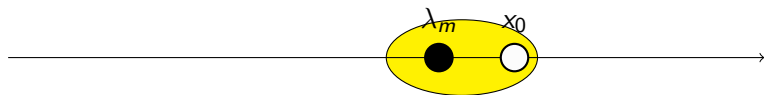
with **bisection** we can find the neighbourhood of λ_m

Example

we can now use the Laguerre's iteration



Example



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 $t \cdot v = \lambda \cdot s \cdot v$. $s \neq 0$, 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

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

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

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 the most 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:

Speudocode:

```
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 #
```

*This is the most 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”)

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

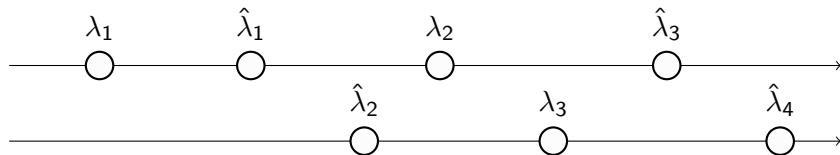
$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

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

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

Remark

It's possible that



Split

Teo (A sort of “interlacing”)

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

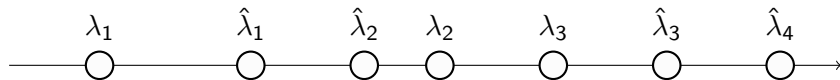
$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

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

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

Remark

It's possible that



Split

Teo (A sort of “interlacing”)

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

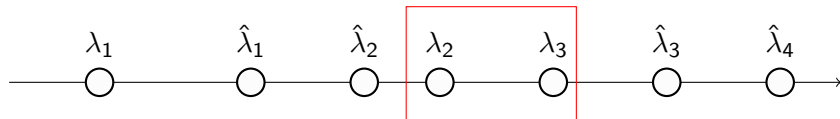
$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

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

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

Remark

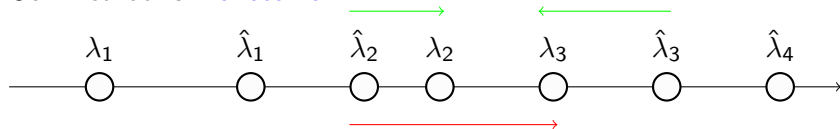
It's possible that



Split

Remark

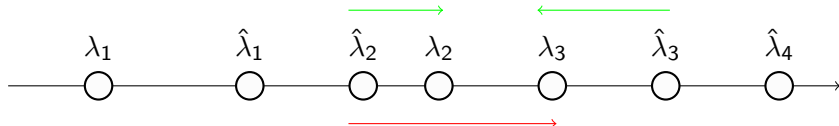
Our method is **monotonic**



Split

Remark

Our method is **monotonic**

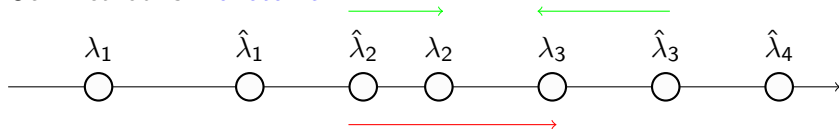


We can reach λ_2 only from $\hat{\lambda}_2$, moving from left to right (and similar λ_3 only from $\hat{\lambda}_3$, moving from right to left).

Split

Remark

Our method is **monotonic**



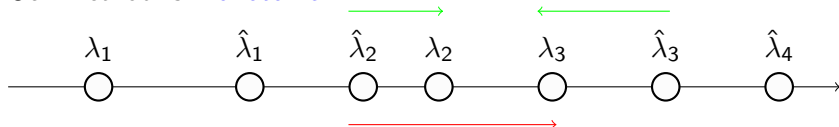
We can reach λ_2 only from $\hat{\lambda}_2$, moving from left to right (and similar λ_3 only from $\hat{\lambda}_3$, moving from right to left).

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).

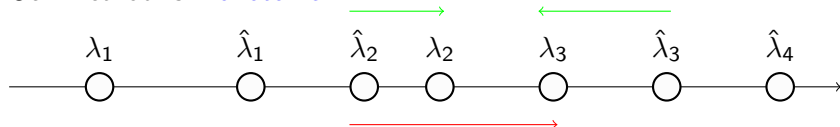
If $|\lambda_2 - \lambda_3| \leq 10^{-14}$ we can have troubles.

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*.

Speudocode:

```
mlt = 1
do k = 1, ...
  m = j + k  $\operatorname{sgn}\left(-\frac{f'(x)}{f(x)}\right)$ 
  if m ≤ 0 then
    something goes wrong
  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
```

Appendice A: proof of Laguerre's convergence

► back

Appendice B: proof of property about Sturm sequence

► back

Appendice C: code

► back

Grazie per l'attenzione.