

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

July 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*

symmetric and tridiagonal

pencil.

Def (Problem)

Find λ such that $Tv = \lambda Sv$.

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

Matrix storin

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

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

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

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

Oss

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

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

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

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

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

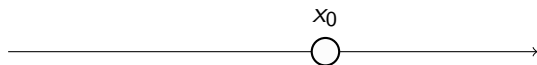
and with algebraic manipulations:

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

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

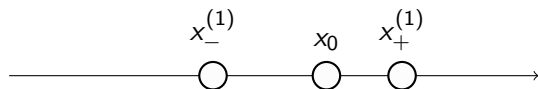
Fast and secure iterative method

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



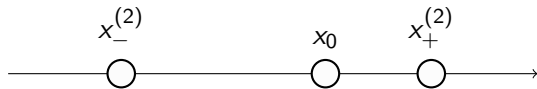
Fast and secure iterative method

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.

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.

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. $\text{sign}(\frac{f'(x_0)}{f(x_0)}) = \text{sign}(\lambda_m - x_0)$ then $\{x_-^{(k)}\}_{k=1,\dots}$ converge monotonically in asymptotically cubic rate to λ_m .*

*for $x_0 < \lambda_{m+1}$ s.t. $\text{sign}(\frac{f'(x_0)}{f(x_0)}) = \text{sign}(\lambda_m - x_0)$ 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. $\text{sign}(\frac{f'(x_0)}{f(x_0)}) = \text{sign}(\lambda_m - x_0)$ 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 **bisection**)

*for $x_0 < \lambda_{m+1}$ s.t. $\text{sign}(\frac{f'(x_0)}{f(x_0)}) = \text{sign}(\lambda_m - x_0)$ 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. $\text{sign}(\frac{f'(x_0)}{f(x_0)}) = \text{sign}(\lambda_m - x_0)$ 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 **bisection**)
The Laguerre’s method is **fast**.

*for $x_0 < \lambda_{m+1}$ s.t. $\text{sign}(\frac{f'(x_0)}{f(x_0)}) = \text{sign}(\lambda_m - x_0)$ 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 [2]) causes no trouble, so the most important 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

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

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

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

Oss

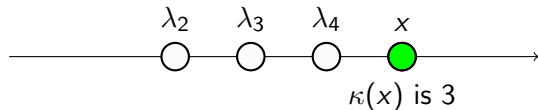
If $x = \lambda$ then $\rho_n(\lambda) = \mathcal{F}_{(T_n, S_n)}(\lambda) = f$

Oss

$\rho_0, \rho_1, \dots, \rho_n$ 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 $\{\rho_i\}_{i=0, \dots, n}$



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

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

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

Ideas:

And if $n = 4$?

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

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

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

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

Split

Teo (A sort of “interlacing”)

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

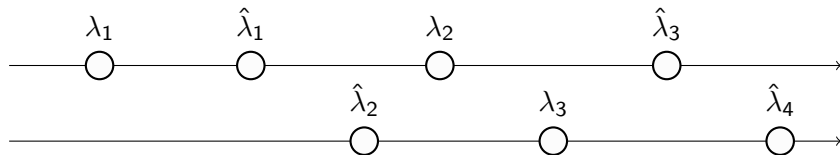
$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

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

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

Oss

It's possible that



Split

Teo (A sort of “interlacing”)

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

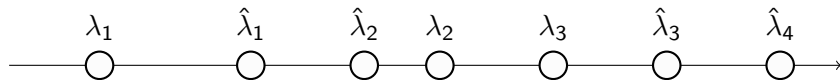
$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

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

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

Oss

It's possible that



Split

Teo (A sort of “interlacing”)

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

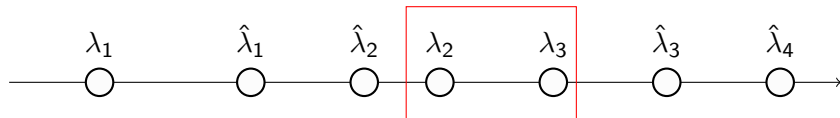
$$\hat{\lambda}_{i-1} \leq \lambda_i \leq \hat{\lambda}_{i+1}$$

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

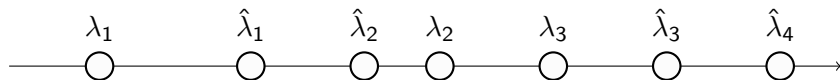
with $i = 2, 3, \dots, n - 1$.

Oss

It's possible that



Split



So we can't

Grazie per l'attenzione.