

EfficientMaxEigenpairTridiagonal Vignette

Mu-Fa Chen

Assisted by Yue-Shuang Li

(Beijing Normal University)

September 25, 2017

EfficientMaxEigenpairTridiagonal is a package for computing the maximal eigenpair of tridiagonal matrices with non-negative off-diagonal elements. This vignette is a simple guide to using the package. All the algorithms and examples provided in this vignette are available in the papers [1] (2016) and [2] (2017) by Mu-Fa Chen.

Let us install and require the package EfficientMaxEigenpairTridiagonal first. We now introduce the package in details. It offers three main algorithms:

- `eff_ini_max eig_tri` ($a, b, c, xi, digit_thresh$): calculate the maximal eigenpair for the tridiagonal matrix based on [1; §3] (2016). The coefficient xi is used in the improved initial z_0 to form a convex combination of $1/\delta_1$ and $(v_0, -Qv_0)_\mu$. Hence, it should be between 0 and 1. The inner product $(\cdot, \cdot)_\mu$ here is in the space $L^2(\mu)$. When xi is a little away from 1, the algorithm becomes more effective but may be more dangerous. The case $xi=1$ is safer but the safest procedure is `eff_ini_max eig_tri1` ($a, b, c, xi, digit_thresh$) given below. For different models, one may choose different xi but need to be cautious to avoid going into pitfall when choosing xi .

–The precise level used for output results is set to be `digit_thresh=6` which implies e-6 without any special requirement. The parameter a, b, c are just the same in the paper [1] (2016), we now state it explicitly. Given tridiagonal matrix A

$$A = \begin{pmatrix} -(b_0 + c_0) & b_0 & 0 & 0 & \cdots \\ a_1 & -(a_1 + b_1 + c_1) & b_1 & 0 & \cdots \\ 0 & a_2 & -(a_2 + b_2 + c_2) & b_2 & \cdots \\ \vdots & \vdots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & a_N & -(a_N + c_N) \end{pmatrix},$$

where a and b should be positive, let $a = (a_1, a_2, \dots, a_N), b = (b_0, b_1, \dots, b_{N-1})$ and $c = (c_0, c_1, \dots, c_N)$. Same for the following two algorithms and the examples shown in this vignette. For `eff_ini_max eig_tri` ($a, b, c, xi, digit_thresh$) and `eff_ini_max eig_1` ($a, b, c, xi, digit_thresh$), c is free but $c_j \neq 0$. By a shifted if necessary, in what follows, we often assume that $c \geq 0$ and $c_j \neq 0$. For `eff_ini_seceig_tri` ($a, b, xi, digit_thresh$), $c_j \equiv 0$.

- `eff_ini_max eig_tri1` ($a, b, c, xi, digit_thresh$): based on [2; §A.4](2017). The safe improvement of [1; §3] (2016).

•`eff_ini_seceig_tri` (a , b , xi , `digit_thresh`): calculate the sub-maximal eigenpair for the tridiagonal matrix. As mentioned in [1] (2016), for simplicity, here we assume that the sums of each row of the input tridiagonal matrix should be 0, i.e. $c_j = 0$ for all $i \in E$. Similarly, there are other options for `xi` and `digit_thresh`, which are the same as defined in the function `eff_ini_maxeig_tri` (a , b , c , xi , `digit_thresh`).

There are six auxiliary functions:

•`tridiag`: generate a tridiagonal matrix A based on three input vectors a , b and free c .

•`eff_ini_tri`: generate the efficient initial eigenpair for the tridiagonal matrix A based on three input vectors a , b , c but $c_j \neq 0$.

•`solu`: explicit representation of the solution w to equation $(-Q - zI)w = v$ based on Q , v , z .

•`ray_quot`: Rayleigh quotient iteration algorithm for computing the maximal eigenpair of matrix Q .

•`delta`: compute δ_k for given v and Q in [2; §A.4] (2017).

•`shif_inv_delta`: shifted inverse iteration algorithm using δ_k to compute the maximal eigenpair of matrix Q .

•`testing`: main function including the following examples.

Examples

In this vignette, we show how to apply the MATLAB procedure to the most of the tridiagonal examples in [1] (2016) and an example in [2] (2017).

The computational efficiency and results of the algorithm are studied as follows. The numerical experiments are fulfilled on a PC with Intel(R) Core(TM) i5-5200 CPU @2.20 GHz and 4.00 GB RAM. These algorithms are implemented in MATLAB(R2013a)

The maximal eigenpair (I)

In this subsection, we show how to apply the procedure to [1; §3] (2016).

The minimal eigenvalue of $-Q$ Example 7 in [1] (2016)

```
n = 7; xi = 1; digit_thresh = 6;
a = (1 : n).^2; b = a; c = zeros(1, n + 1); c(n + 1) = (n + 1)^2;
[z, ~, ~] = eff_ini_maxeig_tri(a, b, c, xi, digit_thresh)
```

Here $z = [z_0, z_1, z_2]$. To save 6 decimal digits, we need only two iterations. When $n = 7, 99, 499, 999, 4999, 7499, 9999$, using the initial $1/\delta_1$ only, the outputs are as follows

```
[0.485985, 0.525313, 0.525268]
[0.348549, 0.376437, 0.376383]
[0.310195, 0.338402, 0.338329]
```

[0.299089, 0.327320, 0.327240]
 [0.281156, 0.308623, 0.308529] (less than 28 seconds)
 [0.277865, 0.305016, 0.304918] (less than 77 seconds)
 [0.275762, 0.302660, 0.302561] (less than 177 seconds)

When $n = 99, 499, 999, 4999, 7499, 9999$, using convex combination for $\xi = 7/8$, the outputs are as follows

[0.387333, 0.376393, 0.376383]
 [0.349147, 0.338342, 0.338329]
 [0.338027, 0.327254, 0.327240]
 [0.319895, 0.308550, 0.308529] (less than 28 min)
 [0.316529, 0.304942, 0.304918] (less than 76 seconds)
 [0.314370, 0.302586, 0.302561] (less than 174 seconds)

The maximal eigenvalue of A Example 8 in [1] (2016)

$a = 14/100$; $b = 40/100$;
 $c = [-25/100 - 40/100, -12/100 - 14/100]$;
 $[z, \sim, \sim] = \text{eff_ini_maxeig_tri}(a, b, c, 1, 6)$
 $[z, \sim, \sim] = \text{eff_ini_maxeig_tri}(a, b, c, 7/8, 6)$

Here $z = [z_0, z_1, z_2]$. For this example, we need a constant shift mI from the original matrix. The outputs are as follows

Input vector c should be all nonnegative! $m = 6.500000e - 01$
 $z = [0.437923, 0.430407, 0.430408]$
 Input vector c should be all nonnegative! $m = 6.500000e - 01$
 $z = [0.436733, 0.430407, 0.430408]$

Example 10 (T) in [1](2016)

$a = [\text{sqrt}(10), \text{sqrt}(130)/11]$; $b = [11/\text{sqrt}(10), 20 * \text{sqrt}(130)/143]$;
 $c = [-1 - 11/\text{sqrt}(10), -25/11 - \text{sqrt}(10) - 20 * \text{sqrt}(130)/143, -8/11 - \text{sqrt}(130)/11]$;
 $[z, \sim, \sim] = \text{eff_ini_maxeig_tri}(a, b, c, 7/8, 6)$
 $[z, \sim, \sim] = \text{eff_ini_maxeig_tri}(a, b, c, 1, 6)$

Here $z = [z_0, z_1, z_2]$. For this example, we need a constant shift m from the original matrix. The outputs are as follows

Input vector c should be all nonnegative! $m = 7.029656e + 00$
 $z = [5.31610, 5.23598, 5.23607]$
 Input vector c should be all nonnegative! $m = 7.029656e + 00$
 $z = [5.37522, 5.23580, 5.23607]$

Example A3 in [2] (2017)

$a = [0.5142, 0.2115, 0.8442, 0.2347, 0.9837]$;
 $b = [0.9962, 0.1111, 0.1405, 0.7595, 0.0781]$;
 $c = [-2.334 - 0.9962, -2.6725 - 0.5142 - 0.1111, -2.263 - 0.2115 - 0.1405,$
 $-2.8457 - 0.8442 - 0.7595, -2.2257 - 0.2347 - 0.0781, -2.1582 - 0.9837]$;

```
[z, ~, iter] = eff_ini_maxeig_tri (a, b, c, 1, 6)
[z, ~, iter] = eff_ini_maxeig_tri (a, b, c, 0, 6)
```

Here $z = [z_0, \dots, z_{iter}]$. For this example, we need a constant shift m from the original matrix. Here we remark that $\xi \neq 0$ for safe, else it may go into pitfall just like the second outputs. The outputs are as follows

```
Input vector c should be all nonnegative! m = 4.449400e + 00
z = [3.35401, 3.26180, 3.26752, 3.26753]
iter = 3
Input vector c should be all nonnegative! m = 4.449400e + 00
z = [3.11543, 3.19197, 3.16652, 3.16287, 3.16251, 3.16247, 3.16247]
iter = 6
```

The maximal eigenpair (II)

In this subsection, we show how to use the procedure from [2; §A.4] (2017). From this subsection, we can see that the procedure `eff_ini_maxeig_tril` may need one more step than procedure `eff_ini_maxeig_tri`, but `eff_ini_maxeig_tril` is safe and never goes to pitfall, so in some sense, `eff_ini_maxeig_tril` is enough for computing the maximal eigenpair of tridiagonal matrix.

The minimal eigenvalue of $-Q$ Example 7 in [1] (2016)

```
n = 7; xi = 1; digit_thresh = 6;
a = (1:n)ˆ2; b = a; c = zeros(1, n + 1); c(n + 1) = (n + 1)ˆ2;
[z, ~, ~] = eff_ini_maxeig_tril (a, b, c, xi, digit_thresh)
```

Here $z = [z_0, z_1, z_2, z_3]$. To save 6 decimal digits, we need only three iterations. When $n = 7, 99, 499, 999, 4999, 7499, 9999$, using the initial $1/\delta_1$ only, the outputs are as follows

```
[0.485985, 0.524150, 0.525267, 0.525268]
[0.348549, 0.374848, 0.376378, 0.376383]
[0.310195, 0.336860, 0.338320, 0.338329]
[0.299089, 0.325735, 0.327229, 0.327240]
[0.281156, 0.306874, 0.308514, 0.308529](less than 25 seconds)
[0.277865, 0.303213, 0.304903, 0.304918](less than 74 seconds)
[0.275762, 0.300821, 0.302545, 0.302561](less than 185 seconds)
```

The maximal eigenvalue of A Example 8 in [1] (2016)

```
a = 14/100; b = 40/100;
c = [-25/100 - 40/100, -12/100 - 14/100];
[z, ~, ~] = eff_ini_maxeig_1 (a, b, c, 1, 6)
[z, ~, ~] = eff_ini_maxeig_1 (a, b, c, 7/8, 6)
```

Here $z = [z_0, z_1, z_2]$. For this example, we need a constant shift mI from the original matrix. The outputs are as follows

```
Input vector c should be all nonnegative!
```

```

m = 6.500000e - 01
z = [0.437923, 0.430519, 0.430408]
Input vector c should be all nonnegative!
m = 6.500000e - 01
z = [0.436733, 0.430502, 0.430408]

```

Example 10 (T) in [1](2016)

```

a = [sqrt(10), sqrt(130)/11]; b = [11/sqrt(10), 20 * sqrt(130)/143];
c = [-1-11/sqrt(10), -25/11-sqrt(10)-20*sqrt(130)/143, -8/11-sqrt(130)/11];
[z, ~, ~] = eff_ini_max eig_tri (a, b, c, 7/8, 6)
[z, ~, ~] = eff_ini_max eig_tri (a, b, c, 1, 6)

```

Here $z = [z_0, z_1, z_2, z_3]$. For this example, we need a constant shift m from the original matrix. The outputs are as follows

```

Input vector c should be all nonnegative! m = 7.029656e + 00
z = [5.31610, 5.23749, 5.23607, 5.23607]
Input vector c should be all nonnegative! m = 7.029656e + 00
z = [5.37522, 5.23853, 5.23607, 5.23607]

```

Example A3 in Chen(2017)

```

a = [0.5142, 0.2115, 0.8442, 0.2347, 0.9837];
b = [0.9962, 0.1111, 0.1405, 0.7595, 0.0781];
c = [-2.334 - 0.9962, -2.6725 - 0.5142 - 0.1111, -2.263 - 0.2115 - 0.1405,
-2.8457 - 0.8442 - 0.7595, -2.2257 - 0.2347 - 0.0781, -2.1582 - 0.9837];
[z, ~, ~] = eff_ini_max eig_tril (a, b, c, 1, 6)

```

Here $z = [z_0, \dots, z_{iter}]$. For this example, we need a constant shift m from the original matrix. Since this procedure guarantees the safeness, it needs one or two more steps to get the results comparing with (I). The outputs are as follows

```

Input vector c should be all nonnegative! m = 4.449400e + 00
z = [3.35401, 3.27947, 3.26850, 3.26754, 3.26753]
iter = 4

```

The sub-maximal eigenpair

Example 25 in [1] (2016)

```

a = (1 : 7).^2; b = a;
[z, ~, ~] = eff_ini_seceig_tri (a, b, 0, 6)
[z, ~, ~] = eff_ini_seceig_tri (a, b, 1, 6)
[z, ~, ~] = eff_ini_seceig_tri (a, b, 2/5, 6)

```

Here $z = [z_0, z_1, z_2]$, the outputs are as follows

```

z = [0.902633, 0.820614, 0.820539]

```

$$z = [0.456343, 0.821600, 0.820539]$$

$$z = [0.724117, 0.820629, 0.820539]$$

Example 26 in Chen(2016)

```

a = [3 2 10 11]; b = [5 4 1 6];
[z, ~, iter] = eff_ini_seceig_tri (a, b, 0, 6 )
[z, ~, iter] = eff_ini_seceig_tri (a, b, 1, 6 )
[z, ~, iter] = eff_ini_seceig_tri (a, b, 2/5, 6)

```

Here $z = [z_0, \dots, z_2]$. The outputs are as follows

$$z = [3.84977, 3.05196, 3.03673]$$

$$z = [1.72924, 3.05715, 3.03673]$$

$$z = [3.00156, 3.03675, 3.03673]$$

Acknowledgments The research project is supported in part by the National Natural Science Foundation of China (Nos. 11131003, 11626245, 11771046) and the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions. The first author acknowledges the assisted one for her hard work in preparing this package in MATLAB.

References

- [1] Chen, M.F. (2016). *Efficient initials for computing the maximal eigenpair*. Front. Math. China 11(6), 1379–1418.
- [2] Chen, M.F. (2017). *Global algorithms for maximal eigenpair*. Frontiers of Mathematics in China, 1–21.

Additional references can be found from volumes 1-4 in the middle of Chen's homepage:

<http://math0.bnu.edu.cn/~chenmf>

Maintainer Yue-Shuang Li, email: liyueshuang at mail.bnu.edu.cn