

EfficientMaxEigenpairGeneral Vignette

Mu-Fa Chen

Assisted by Yue-Shuang Li

(Beijing Normal University)

September 30, 2017

EfficientMaxEigenpairGeneral is a package for computing the maximal eigenpair of general matrices with non-negative off-diagonal elements especially for tridiagonally dominant matrices. This vignette is a simple guide to using the package. The algorithm 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 EfficientMaxEigenpairGeneral first. We now introduce the package in details.

- `eff_ini_maxeig_general` (A , ξ , `digit_thresh`): calculate the maximal eigenpair for the general matrix with non-negative off-diagonal elements based on [1; §4] (2016). The coefficient ξ is used to choose initials, $\xi = 1$ denotes $z_0 = 1/\delta_1$ and $\xi = 0$ denotes $z_0 = (v_0, -Qv_0)_\mu$. Hence, it should be between 0 and 1. When ξ is close to 1, it becomes relatively safer (or as a dual, when ξ is close to 0, it becomes more dangerous). The inner product $(\cdot)_\mu$ here is in the space $L^2(\mu)$. This procedure is effective for tridiagonally dominant matrices.

- The precise level used for output results is set to be `digit_thresh=6` which implies e-6 without any special requirement.

There is an auxiliary function for the package:

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

There is a test for the examples in this vignette:

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

Examples

In this vignette, we show how to apply the MATLAB procedure to most of the general examples in [1; §4] (2016) and two examples in [2; §2] (2017). Here some outputs are different from the original paper is due to the fact that we adopt the revised version.

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)

Example 18 (Same as Example 10) in [1](2016)

```

digit_thresh= 6;
A = [1 2 3; 1 2 1; 3 2 1];
[z, ~, ~] = eff_ini_maxeig_general (A, 1, digit_thresh)
[z, ~, ~] = eff_ini_maxeig_general (A, 1/3, digit_thresh)
[z, ~, ~] = eff_ini_maxeig_general (A, 0, digit_thresh)
Here

```

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 1 \\ 3 & 2 & 1 \end{pmatrix},$$

$z = [z_0, z_1, z_2]$. To save 6 decimal digits, we need only two iterations using the initials $1/\delta_1$ and $(v_0, -Qv_0)_\mu$ respectively. For this example, we need a constant shift mI from the original matrix, the outputs are as follows

```

m = 6
z = [5.90218, 5.23550, 5.23607]
m = 6
z = [5.43571, 5.23601, 5.23607]
m = 6
z = [5.20248, 5.23607, 5.23607]

```

Example 19(Same as Example 14) in [1] (2016)

```

A = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16];
[z, ~, iter] = eff_ini_maxeig_general (A, 1, 6)
[z, ~, iter] = eff_ini_maxeig_general (A, 1/3, 6)
[z, ~, iter] = eff_ini_maxeig_general (A, 0, 6)

```

Here $z = [z_0, z_1, \dots, z_{\text{iter}}]$. For this example, we need a constant shift mI from the original matrix. The outputs are as follows

```

m = 58
z = [57.6110, 35.4882, 36.2081, 36.2094]   iter= 3
m = 58
z = [40.0812, 36.1690, 36.2094]   iter= 2
m = 58
z = [31.3163, 36.0541, 36.2095, 36.2094]   iter= 3

```

Example 20 (Same as Example 15) in [1](2016)

```

A = [1 2 0 0; 3 14 11 0; 9 10 11 1; 5 6 7 8]
[z, ~, iter] = eff_ini_maxeig_general (A, 1, 6)
[z, ~, iter] = eff_ini_maxeig_general (A, 0.65, 6)
[z, ~, iter] = eff_ini_maxeig_general (A, 0, 6)

```

Here $z = [z_0, z_1, \dots, z_{\text{iter}}]$. For this example, we need a constant shift m from the original matrix. The outputs are as follows

```

m = 31

```

```

z = [30.6865, 23.6432, 24.0257, 24.0293]  iter = 3
m = 31
[26.5804, 23.9641, 24.0290, 24.0293]  iter = 3
m = 31
z = [18.9548, 23.0428, 24.0437, 24.0293, 24.0293]  iter = 4

```

Example 21 in [1] (2016)

```

b4 = 0.01
Q = [-3, 2, 0, 1, 0; 4, -7, 3, 0, 0; 0, 5, -5, 0, 0; 10, 0, 0, -16, 6; 0, 0, 0, 11, -11-b4];
[z, ~, iter] = eff_ini_max eig_general (Q, 1, 6)
[z, ~, iter] = eff_ini_max eig_general (Q, 0, 6)

```

Here $z = [z_0, \dots, z_{iter}]$. The outputs are as follows

```

z = [-0.000278785] (xi= 0, b = 0.01)  iter = 0
z = [0.227232 * 10-15, -0.000278686] (xi= 1, b = 0.01)  iter = 1
z = [-0.0253418, -0.0245175] (xi= 0, b = 1)  iter = 1
z = [0.229215 * 10-15, -0.0245176] (xi= 1, b = 1)  iter = 1
z = [-0.305870, -0.182834, -0.182819] (xi= 0, b = 100)  iter = 2
z = [0.239635 * 10-15, -0.182847, -0.182819] (xi= 1, b = 100)  iter = 2
z = [-0.424900, -0.195207, -0.195145] (xi= 0, b = 106)  iter = 2
z = [0.240125 * 10-15, -0.195180, -0.195145] (xi= 1, b = 106)  iter = 2

```

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, 2.6725, 2.263, 2.8457, 2.2257, 2.1582];
A = diag(c) + diag(a, -1) + diag(b, 1);
[z, ~, iter] = eff_ini_max eig_general (A, 1, 6)
[z, ~, iter] = eff_ini_max eig_general (A, 0.18, 6)
[z, ~, iter] = eff_ini_max eig_general (A, 0, 6)

```

Here $z = [z_0, \dots, z_{iter}]$. The outputs are as follows

```

m = 4.449400e + 00
[3.93158, 3.22875, 3.26172, 3.26751, 3.26753]  iter= 4
m = 4.449400e + 00
[3.26234, 3.26746, 3.26753]  iter= 2
m = 4.449400e + 00
[3.11543, 3.19197, 3.16652, 3.16287, 3.16251, 3.16247]  iter= 5

```

From this example, we can see that when $xi=0$, it goes into pitfall. So we may choose $xi \neq 0$.

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