PowerfulTriMaxEigenpair Vignette

Yue-Shuang Li
Xiao-Jun Mao

2019-05-19

`PowerfulTriMaxEigenpair` is a package for computing the maximal eigenpair of Hermitizable tridiagonal matrices. This vignette is a simple guide to using the package. All the algorithms and examples provided in this vignette are available in the paper "Speed of stability for birth–death processes" and "Hermitizable, isospectral complex matrices or differential operators" by Mu-Fa Chen and "Development of powerful algorithm for maximal eigenpair" by Mu-Fa Chen and Yue-Shuang Li. The papers Chen (2010) and Chen (2018) are now included in the Vol 4, in the middle of the website:

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

Let us install and require the package `PowerfulTriMaxEigenpair` first.

```
require(Powerful_MaxEigenpair)
```

It offers two main algorithms:

- `powerful.maxeig.tri()`: calculate the maximal eigenpair for Hermitizable tridiagonal matrix.
  - The precise level used for output results is set to be `digit.thresh = 6` which implies 1e-6 without any special requirement. Same for the following three algorithms and the examples shown in this vignette.
  - This algorithm works only for Hermitizable tridiagonal matrix.
- `powerful.secondeig.tri()`: calculate the next to maximal eigenpair for the tridiagonal matrix. As mentioned in the cited paper, for simplicity, here we assume that the sums of each row of the input tridiagonal matrix should be 0, i.e, $A_i = 0$ for all $i \in E$. Similarly, there is an option `digit.thresh` which is the same as defined in function `powerful.maxeig.tri()`.

There are two auxiliary functions `tridia()` and `thomas.tri.sol()` where:

- `tridiag()`: generate tridiagonal matrix A based on three input vectors.
- `thomas.tri.sol()`: solve the linear equation $(-Q-z*I)w=v$ by Thomas algorithm.

Acknowledgement

The algorithm is reformulated based on a series of recent papers by Mu-Fa Chen. The research project is supported in part by the National Natural Science Foundation of China (No. 11131003, 11626245, 11771046) and the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

Examples

We show most of the examples in the paper "Development of powerful algorithm for maximal eigenpair" by Mu-Fa Chen and Yue-Shuang Li in this section. For a convenient comparison, we keep the same subsection names and examples as the paper "Development of powerful algorithm for maximal eigenpair".

The maximal eigenpair

Armed with `powerful.maxeig.tri()`, we can calculate the maximal eigenpair for the tridiagonal matrix.

Example 4

```
nn = c(8, 16, 50, 100, 500, 1000, 5000, 10000, 15000)
for (j in 1:9) {
    a = rep(1, (nn[j] - 1))
    b = rep(2, (nn[j] - 1))
    C = rep(3, nn[j])

    if (j <= 4) {
        print(powerful.maxeig.tri(a, b, C, digit.thresh = 6)$z[1:5])
    } else {
        if (j <= 6) {
            print(powerful.maxeig.tri(a, b, C, digit.thresh = 6)$z[1:3])
        } else {
            print(powerful.maxeig.tri(a, b, C, digit.thresh = 6)$z[1:2])
        }
    }
}
```

```
## [1] 0.253835 0.335440 0.342107 0.342148 0.342148
## [1] 0.182046 0.215330 0.219673 0.219732 0.219732
## [1] 0.171577 0.175993 0.176912 0.176937 0.176937
## [1] 0.171573 0.172686 0.172934 0.172941 0.172941
## [1] 0.171573 0.171618 0.171628
## [1] 0.171573 0.171584 0.171587
## [1] 0.171573 0.171573
## [1] 0.171573 0.171573
## [1] 0.171573 0.171573
```

Example 5

```
a = c(0.5142, 0.2115, 0.8442, 0.2347, 0.9837)
b = c(0.9962, 0.1111, 0.1405, 0.7595, 0.0781)
C = c(-2.334, -2.6725, -2.263, -2.8457, -2.2257, -2.1582)
maxeig = powerful.maxeig.tri(a, b, C, digit.thresh = 6)

print(maxeig$m - maxeig$z[1:5])
```

```
## [1] 3.414012 3.287209 3.269574 3.267574 3.267534
```

Example 6

```
a = as.complex(c(4 * 2 - (0+4i), 4 * 3 - (0+4i), 1 - (0+2i), 2 - (0+3i), 4 +
    (0+2i)))
b = as.complex(c(2 + (0+1i), 3 + (0+1i), 1 + (0+2i), 2 + (0+3i), 2 - (0+1i)))
C = c(2, 1, 3, 2, 4, -3)
maxeig = powerful.maxeig.tri(a, b, C, digit.thresh = 6)

print(maxeig$m - maxeig$z[1:5])
```

```
## [1] 7.315933 6.342822 6.203820 6.196437 6.196404
```

Example 7

```
a = as.complex(c(4 * 2 - (0+4i), 36 * 2 - (0+36i), 144 * 2 - (0+144i), 400 *
    2 - (0+400i)))
b = as.complex(c(2 + (0+1i), 32 + (0+16i), 81 * 2 + (0+81i), 256 * 2 + (0+256i)))
C = c(1, 1, 1, 1, 1)
maxeig = powerful.maxeig.tri(a, b, C, digit.thresh = 6)

print(maxeig$m - maxeig$z[1:3])
```

```
## [1] 773.8360 754.5937 754.3913
```

Example 8

```
nn = c(8, 16, 32, 50, 100)
for (i in 1:5) {
    a = c(1:(nn[i] - 1))
    b = 2 * c(1:(nn[i] - 1))
    C = c(b[1], 3 * c(1:(nn[i] - 1)) + 2)

    print(powerful.maxeig.tri(a, b, C, digit.thresh = 5)$z[1:4])
}
```

```
## [1] 0.83604 0.99214 1.01307 1.01355
## [1] 0.83404 0.97704 0.99939 1.00011
## [1] 0.83404 0.97665 0.99925 1.00000
## [1] 0.83404 0.97665 0.99925 1.00000
## [1] 0.83404 0.97665 0.99925 1.00000
```

Example 9

```
nn = c(8, 50, 100, 500, 1000, 5000)
for (i in 1:6) {
    a = c(1:(nn[i] - 1))
    a = a * (a - 1/2) * (a^2 + 3 * a + 3)
    b = c(1:(nn[i] - 1))^4
    C = c(b[1], a[1:(nn[i] - 2)] + b[2:(nn[i] - 1)], a[nn[i] - 1] + nn[i]^4)

    print(powerful.maxeig.tri(a, b, C, digit.thresh = 6)$z[1:4])
}
```

```
## [1] 0.416918 0.627353 0.633461 0.633466
## [1] 0.357080 0.541483 0.548162 0.548169
## [1] 0.347301 0.526674 0.533345 0.533353
## [1] 0.335057 0.507877 0.514489 0.514498
## [1] 0.332284 0.503583 0.510173 0.510182
## [1] 0.328655 0.497945 0.504504 0.504512
```

Example 10

```
nn = c(8, 100, 500, 1000, 5000, 7500, 10^4)
for (i in 1:7) {
    a = c(1:(nn[i] - 1))^2
    b = c(1:(nn[i] - 1))^2
    C = c(b[1], a[1:(nn[i] - 2)] + b[2:(nn[i] - 1)], a[nn[i] - 1] + nn[i]^2)
```

```
    print(powerful.maxeig.tri(a, b, C, digit.thresh = 6)$z[1:4])
}
```

```
## [1] 0.406762 0.514094 0.525176 0.525268
## [1] 0.304993 0.369950 0.376269 0.376383
## [1] 0.279999 0.333226 0.338230 0.338329
## [1] 0.273336 0.322412 0.327148 0.327240
## [1] 0.263220 0.304128 0.308454 0.308529
## [1] 0.261484 0.300603 0.304845 0.304918
## [1] 0.260397 0.298305 0.302489 0.302561
```

Example 11

```
nn = c(0.01, 1, 100, 10^6)
a = c(3, 2, 10, 11)
b = c(5, 4, 1, 6)

for (i in 1:4) {
    C = c(5, 7, 3, 16, 11 + nn[i])

    if (i == 1) {
        print(powerful.maxeig.tri(a, b, C, digit.thresh = 9)$z[1:4])
    }

    if (i == 2) {
        print(powerful.maxeig.tri(a, b, C, digit.thresh = 7)$z[1:4])
    }

    if (i >= 3) {
        print(powerful.maxeig.tri(a, b, C, digit.thresh = 6)$z[1:4])
    }
}
```

```
## [1] 0.000143394 0.000278683 0.000278686 0.000278686
## [1] 0.0130396 0.0244922 0.0245175 0.0245175
## [1] 0.102368 0.182367 0.182819 0.182819
## [1] 0.109962 0.194680 0.195145 0.195145
```

Next to the maximal eigenpair

Armed with `powerful.seceig.tri()`, we can calculate next to the maximal eigenpair for the tridiagonal matrix.

Example 14

```
nn = c(8, 16, 50, 100, 1000, 10000, 15000)
for (j in 1:7) {

    a = rep(1, (nn[j] - 1))
    b = rep(2, (nn[j] - 1))

    if (j <= 5) {
        print(powerful.seceig.tri(a, b, digit.thresh = 6)$z[1:4])
```

```
    } else {
        print(powerful.seceig.tri(a, b, digit.thresh = 6)$z[1:2])
    }

}
```

```
## [1] 0.284681 0.379670 0.386839 0.386874
## [1] 0.184834 0.221395 0.225864 0.225920
## [1] 0.171578 0.176176 0.177128 0.177154
## [1] 0.171573 0.172709 0.172961 0.172969
## [1] 0.171573 0.171584 0.171587 0.171587
## [1] 0.171573 0.171573
## [1] 0.171573 0.171573
```

Example 15

```
nn = c(8, 16, 50, 100, 1000, 10000)
for (i in 1:6) {
    a = c(1:(nn[i] - 1))^2
    b = c(1:(nn[i] - 1))^2

    print(powerful.seceig.tri(a, b, digit.thresh = 6)$z[1:4])
}
```

```
## [1] 0.604490 0.803180 0.820402 0.820539
## [1] 0.481611 0.638168 0.650021 0.650141
## [1] 0.379585 0.494848 0.503500 0.503596
## [1] 0.344814 0.444154 0.451895 0.451977
## [1] 0.287724 0.354814 0.361160 0.361228
## [1] 0.266398 0.315346 0.320907 0.320976
```

Example 16

```
a = c(3, 2, 10, 11)
b = c(5, 4, 1, 6)

print(powerful.seceig.tri(a, b, digit.thresh = 5)$z[1:4])
```

```
## [1] 2.48980 2.97585 3.03569 3.03673
```

References

[1] M. F. Chen. "Efficient initials for computing the maximal eigenpair."" In: *Frontiers of Mathematics in China* 11(6) (2016), pp. 1379–1418.

[2] M. F. Chen. "Global algorithms for maximal eigenpair."" In: *Frontiers of Mathematics in China* 12(5) (2017), pp. 1023–1043.

[3] M. F. Chen. "Hermitizable, isospectral complex matrices or differential operators." In: *Frontiers of Mathematics in China* 13(6) (2018), pp. 1267-1311.

[4] M. F. Chen. and Y.S. Li. "Development of powerful algorithm for maximal eigenpair." In: *Frontiers of Mathematics in China* (2018)

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

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