Package ‘EfficientMaxEigenpair’

January 20, 2025

Type Package

Title Efficient Initials for Computing the Maximal Eigenpair

Version 0.1.4

Date 2017-10-17

Author Mu-Fa Chen <mfchen@bnu.edu.cn>

Maintainer Xiao-Jun Mao <maoxj.ki@gmail.com>

Description An implementation for using efficient initials to compute the maximal eigenpair in R. It provides three algorithms to find the efficient initials under two cases: the tridiagonal matrix case and the general matrix case. Besides, it also provides two algorithms for the next to the maximal eigenpair under these two cases.

License MIT + file LICENSE

URL http://github.com/mxjki/EfficientMaxEigenpair

BugReports http://github.com/mxjki/EfficientMaxEigenpair/issues

Depends R (>= 3.3.2), stats

Encoding UTF-8

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2017-10-23 12:00:48 UTC

Contents

`eff.ini.maxeig.general` *General matrix maximal eigenpair*

Description

Calculate the maximal eigenpair for the general matrix.

Usage

```
eff.ini.maxeig.general(A, v0_tilde = NULL, z0 = NULL, z0numeric, xi = 1,
  digit.thresh = 6)
```

Arguments

`A`	The input general matrix.
`v0_tilde`	The unnormalized initial vector $\tilde{v0}$.
`z0`	The type of initial z_0 used to calculate the approximation of $\rho(Q)$. There are three types: 'fixed', 'Auto' and 'numeric' corresponding to three choices of z_0 in paper.
`z0numeric`	The numerical value assigned to initial z_0 as an approximation of $\rho(Q)$ when z_0='numeric'.
`xi`	The coefficient used to form the convex combination of δ_1^{-1} and $(v_0, -Q * v_0)_\mu$, it should between 0 and 1.
`digit.thresh`	The precise level of output results.

Value

A list of eigenpair object are returned, with components z, v and $iter$.

`z`	The approximating sequence of the maximal eigenvalue.
`v`	The approximating eigenfunction of the corresponding eigenvector.
`iter`	The number of iterations.

See Also

eff.ini.maxeig.tri for the tridiagonal matrix maximal eigenpair by rayleigh quotient iteration algorithm. eff.ini.maxeig.shift.inv.tri for the tridiagonal matrix maximal eigenpair by shifted inverse iteration algorithm.

Examples

```
A = matrix(c(1, 1, 3, 2, 2, 2, 3, 1, 1), 3, 3)
eff.ini.maxeig.general(A, v0_tilde = rep(1, dim(A)[1]), z0 = 'fixed')

A = matrix(c(1, 1, 3, 2, 2, 2, 3, 1, 1), 3, 3)
eff.ini.maxeig.general(A, v0_tilde = rep(1, dim(A)[1]), z0 = 'Auto')

##Symmetrizing A converge to second largest eigenvalue
A = matrix(c(1, 3, 9, 5, 2, 14, 10, 6, 0, 11, 11, 7, 0, 0, 1, 8), 4, 4)
S = (t(A) + A)/2
N = dim(S)[1]
a = diag(S[-1, -N])
b = diag(S[-N, -1])
c = rep(NA, N)
c[1] = -diag(S)[1] - b[1]
c[2:(N - 1)] = -diag(S)[2:(N - 1)] - b[2:(N - 1)] - a[1:(N - 2)]
c[N] = -diag(S)[N] - a[N - 1]

z0ini = eff.ini.maxeig.tri(a, b, c, xi = 7/8)$z[1]
eff.ini.maxeig.general(A, v0_tilde = rep(1, dim(A)[1]), z0 = 'numeric',
z0numeric = 28 - z0ini)
```

---

eff.ini.maxeig.shift.inv.tri

Tridiagonal matrix maximal eigenpair

Description

Calculate the maximal eigenpair for the tridiagonal matrix by shifted inverse iteration algorithm.

Usage

```
eff.ini.maxeig.shift.inv.tri(a, b, c, xi = 1, digit.thresh = 6)
```

Arguments

a	The lower diagonal vector.
b	The upper diagonal vector.
c	The shifted main diagonal vector. The corresponding unshift diagonal vector is -c(b[1] + c[1], a[1:N - 1] + b[2:N] + c[2:N], a[N] + c[N + 1]) where N+1 is the dimension of matrix.

xi	The coefficient used to form the convex combination of δ_1^{-1} and $(v_0, -Q * v_0)_\mu$, it should between 0 and 1.
digit.thresh	The precise level of output results.

Value

A list of eigenpair object are returned, with components z, v and $iter$.

z	The approximating sequence of the maximal eigenvalue.
v	The approximating eigenfunction of the corresponding eigenvector.
iter	The number of iterations.

See Also

`eff.ini.maxeig.tri` for the tridiagonal matrix maximal eigenpair by rayleigh quotient iteration algorithm. `eff.ini.maxeig.general` for the general matrix maximal eigenpair.

Examples

```
a = c(1:7)^2
b = c(1:7)^2
c = rep(0, length(a) + 1)
c[length(a) + 1] = 8^2
eff.ini.maxeig.shift.inv.tri(a, b, c, xi = 1)
```

---

eff.ini.maxeig.tri *Tridiagonal matrix maximal eigenpair*

Description

Calculate the maximal eigenpair for the tridiagonal matrix by rayleigh quotient iteration algorithm.

Usage

```
eff.ini.maxeig.tri(a, b, c, xi = 1, digit.thresh = 6)
```

Arguments

a	The lower diagonal vector.
b	The upper diagonal vector.
c	The shifted main diagonal vector. The corresponding unshift diagonal vector is -c(b[1] + c[1], a[1:N - 1] + b[2:N] + c[2:N], a[N] + c[N + 1]) where N+1 is the dimension of matrix.
xi	The coefficient used to form the convex combination of δ_1^{-1} and $(v_0, -Q * v_0)_\mu$, it should between 0 and 1.
digit.thresh	The precise level of output results.

Value

A list of eigenpair object are returned, with components z, v and $iter$.

`z`	The approximating sequence of the maximal eigenvalue.
`v`	The approximating eigenfunction of the corresponding eigenvector.
`iter`	The number of iterations.

See Also

`eff.ini.maxeig.shift.inv.tri` for the tridiagonal matrix maximal eigenpair by shifted inverse iteration algorithm. `eff.ini.maxeig.general` for the general matrix maximal eigenpair.

Examples

```
a = c(1:7)^2
b = c(1:7)^2
c = rep(0, length(a) + 1)
c[length(a) + 1] = 8^2
eff.ini.maxeig.tri(a, b, c, xi = 1)
```

`eff.ini.seceig.general` *General conservative matrix maximal eigenpair*

Description

Calculate the next to maximal eigenpair for the general conservative matrix.

Usage

```
eff.ini.seceig.general(Q, z0 = NULL, c1 = 1000, digit.thresh = 6)
```

Arguments

`Q`	The input general matrix.
`z0`	The type of initial z_0 used to calculate the approximation of $\rho(Q)$. There are two types: 'fixed' and 'Auto' corresponding to two choices of z_0 in paper.
`c1`	A large constant.
`digit.thresh`	The precise level of output results.

Value

A list of eigenpair object are returned, with components z, v and $iter$.

`z`	The approximating sequence of the maximal eigenvalue.
`v`	The approximating eigenfunction of the corresponding eigenvector.
`iter`	The number of iterations.

Note

The conservativity of matrix $Q = (q_{ij})$ means that the sums of each row of matrix Q are all 0.

See Also

eff.ini.seceig.tri for the tridiagonal matrix next to the maximal eigenpair.

Examples

```
Q = matrix(c(-30, 1/5, 11/28, 55/3291, 30, -17, 275/42, 330/1097,
0, 84/5, -20, 588/1097, 0, 0, 1097/84, -2809/3291), 4, 4)
eff.ini.seceig.general(Q, z0 = 'Auto', digit.thresh = 5)
eff.ini.seceig.general(Q, z0 = 'fixed', digit.thresh = 5)
```

---

eff.ini.seceig.tri *Tridiagonal matrix next to the maximal eigenpair*

Description

Calculate the next to maximal eigenpair for the tridiagonal matrix whose sums of each row should be 0.

Usage

```
eff.ini.seceig.tri(a, b, xi = 1, digit.thresh = 6)
```

Arguments

a	The lower diagonal vector.
b	The upper diagonal vector.
xi	The coefficient used in the improved initials to form the convex combination of δ_1^{-1} and $(v_0, -Q * v_0)_\mu$, it should between 0 and 1.
digit.thresh	The precise level of output results.

Value

A list of eigenpair object are returned, with components z, v and $iter$.

z	The approximating sequence of the maximal eigenvalue.
v	The approximating eigenfunction of the corresponding eigenvector.
iter	The number of iterations.

Note

The sums of each row of the input tridiagonal matrix should be 0.

See Also

eff.ini.seceig.general for the general conservative matrix next to the maximal eigenpair.

Examples

```
a = c(1:7)^2
b = c(1:7)^2

eff.ini.seceig.tri(a, b, xi = 0)
eff.ini.seceig.tri(a, b, xi = 1)
eff.ini.seceig.tri(a, b, xi = 2/5)
```

EfficientMaxEigenpair *EfficientMaxEigenpair: A package for computating the maximal eigenpair for a matrix.*

Description

The EfficientMaxEigenpair package provides some auxillary functions and five categories of important functions: tridiag, tri.sol, find_deltak, ray.quot.tri, shift.inv.tri, ray.quot.seceig.tri, ray.quot.general, ray.quot.seceig.general, eff.ini.maxeig.tri, eff.ini.maxeig.shift.inv.tri, eff.ini.maxeig.general, eff.ini.seceig.tri and eff.ini.seceig.general.

EfficientMaxEigenpair functions

tridiag: generate tridiagonal matrix Q based on three input vectors.

tri.sol: construct the solution of linear equation (-Q-zI)w=v.

find_deltak: compute δ_k for given vector v and matrix Q.

ray.quot.tri: rayleigh quotient iteration algorithm to computing the maximal eigenpair of tridiagonal matrix Q.

shift.inv.tri: shifted inverse iteration algorithm to computing the maximal eigenpair of tridiagonal matrix Q.

ray.quot.seceig.tri: rayleigh quotient iteration algorithm to computing the next to maximal eigenpair of tridiagonal matrix Q.

ray.quot.general: rayleigh quotient iteration algorithm to computing the maximal eigenpair of general matrix A.

ray.quot.seceig.general: rayleigh quotient iteration algorithm to computing the next to maximal eigenpair of general matrix A.

eff.ini.maxeig.tri: calculate the maximal eigenpair for the tridiagonal matrix by rayleigh quotient iteration algorithm.

eff.ini.maxeig.shift.inv.tri: calculate the maximal eigenpair for the tridiagonal matrix by shifted inverse iteration algorithm.

eff.ini.maxeig.general: calculate the maximal eigenpair for the general matrix.

eff.ini.seceig.tri: calculate the next to maximal eigenpair for the tridiagonal matrix whose sums of each row should be 0.

eff.ini.seceig.general: calculate the next to maximal eigenpair for the general conservative matrix.

---

find_deltak *Compute δ_k*

Description

Compute δ_k for given vector v and matrix Q.

Usage

```
find_deltak(Q, v)
```

Arguments

Q	The given tridiagonal matrix.
v	The column vector on the right hand of equation.

Value

A list of δ_k for given vector v and matrix Q.

Examples

```
a = c(1:7)^2
b = c(1:7)^2
c = rep(0, length(a) + 1)
c[length(a) + 1] = 8^2
N = length(a)
Q = tridiag(b, a, -c(b[1] + c[1], a[1:N - 1] + b[2:N] + c[2:N], a[N] + c[N + 1]))
find_deltak(Q, v=rep(1,dim(Q)[1]))
```

---

ray.quot.general *Rayleigh quotient iteration*

Description

Rayleigh quotient iteration algorithm to computing the maximal eigenpair of general matrix A.

Usage

```
ray.quot.general(A, mu, v0_tilde, zstart, digit.thresh = 6)
```

Arguments

`A`	The input matrix to find the maximal eigenpair.
`mu`	A vector.
`v0_tilde`	The unnormalized initial vector $\tilde{v0}$.
`zstart`	The initial z_0 as an approximation of $\rho(Q)$.
`digit.thresh`	The precise level of output results.

Value

A list of eigenpair object are returned, with components z, v and $iter$.

`z`	The approximating sequence of the maximal eigenvalue.
`v`	The approximating eigenfunction of the corresponding eigenvector.
`iter`	The number of iterations.

Examples

```
A = matrix(c(1, 1, 3, 2, 2, 2, 3, 1, 1), 3, 3)
ray.quot.general(A, mu=rep(1,dim(A)[1]), v0_tilde=rep(1,dim(A)[1]), zstart=6,
 digit.thresh = 6)
```

`ray.quot.seceig.general` *Rayleigh quotient iteration*

Description

Rayleigh quotient iteration algorithm to computing the maximal eigenpair of matrix Q.

Usage

```
ray.quot.seceig.general(Q, mu, v0_tilde, zstart, digit.thresh = 6)
```

Arguments

`Q`	The input matrix to find the maximal eigenpair.
`mu`	A vector.
`v0_tilde`	The unnormalized initial vector $\tilde{v0}$.
`zstart`	The initial z_0 as an approximation of $\rho(Q)$.
`digit.thresh`	The precise level of output results.

Value

A list of eigenpair object are returned, with components z, v and $iter$.

z	The approximating sequence of the maximal eigenvalue.
v	The approximating sequence of the corresponding eigenvector.
iter	The number of iterations.

Examples

```
Q = matrix(c(1, 1, 3, 2, 2, 2, 3, 1, 1), 3, 3)
ray.quot.seceig.general(Q, mu=rep(1,dim(Q)[1]), v0_tilde=rep(1,dim(Q)[1]), zstart=6,
 digit.thresh = 6)
```

---

ray.quot.seceig.tri *Rayleigh quotient iteration for Tridiagonal matrix*

Description

Rayleigh quotient iteration algorithm to computing the next to maximal eigenpair of tridiagonal matrix Q.

Usage

```
ray.quot.seceig.tri(Q, mu, v0_tilde, zstart, digit.thresh = 6)
```

Arguments

Q	The input matrix to find the maximal eigenpair.
mu	A vector.
v0_tilde	The unnormalized initial vector $\tilde{v0}$.
zstart	The initial z_0 as an approximation of $\rho(Q)$.
digit.thresh	The precise level of output results.

Value

A list of eigenpair object are returned, with components z, v and $iter$.

z	The approximating sequence of the maximal eigenvalue.
v	The approximating eigenfunction of the corresponding eigenvector.
iter	The number of iterations.

Examples

```
a = c(1:7)^2
b = c(1:7)^2
c = rep(0, length(a) + 1)
c[length(a) + 1] = 8^2
N = length(a)
Q = tridiag(b, a, -c(b[1] + c[1], a[1:N - 1] + b[2:N] + c[2:N], a[N] + c[N + 1]))
ray.quot.seceig.tri(Q, mu=rep(1,dim(Q)[1]), v0_tilde=rep(1,dim(Q)[1]), zstart=6,
 digit.thresh = 6)
```

---

ray.quot.tri — *Rayleigh quotient iteration for Tridiagonal matrix*

Description

Rayleigh quotient iteration algorithm to computing the maximal eigenpair of tridiagonal matrix Q.

Usage

```
ray.quot.tri(Q, mu, v0_tilde, zstart, digit.thresh = 6)
```

Arguments

Q	The input matrix to find the maximal eigenpair.
mu	A vector.
v0_tilde	The unnormalized initial vector $\tilde{v0}$.
zstart	The initial z_0 as an approximation of $\rho(Q)$.
digit.thresh	The precise level of output results.

Value

A list of eigenpair object are returned, with components z, v and $iter$.

z	The approximating sequence of the maximal eigenvalue.
v	The approximating eigenfunction of the corresponding eigenvector.
iter	The number of iterations.

Examples

```
a = c(1:7)^2
b = c(1:7)^2
c = rep(0, length(a) + 1)
c[length(a) + 1] = 8^2
N = length(a)
Q = tridiag(b, a, -c(b[1] + c[1], a[1:N - 1] + b[2:N] + c[2:N], a[N] + c[N + 1]))
ray.quot.tri(Q, mu=rep(1,dim(Q)[1]), v0_tilde=rep(1,dim(Q)[1]), zstart=6,
 digit.thresh = 6)
```

shift.inv.tri *Shifted inverse iteration algorithm for Tridiagonal matrix*

Description

Shifted inverse iteration algorithm algorithm to computing the maximal eigenpair of tridiagonal matrix Q.

Usage

```
shift.inv.tri(Q, mu, v0_tilde, zstart, digit.thresh = 6)
```

Arguments

`Q`	The input matrix to find the maximal eigenpair.
`mu`	A vector.
`v0_tilde`	The unnormalized initial vector $\tilde{v0}$.
`zstart`	The initial z_0 as an approximation of $\rho(Q)$.
`digit.thresh`	The precise level of output results.

Value

A list of eigenpair object are returned, with components z, v and $iter$.

`z`	The approximating sequence of the maximal eigenvalue.
`v`	The approximating eigenfunction of the corresponding eigenvector.
`iter`	The number of iterations.

Examples

```
a = c(1:7)^2
b = c(1:7)^2
c = rep(0, length(a) + 1)
c[length(a) + 1] = 8^2
N = length(a)
Q = tridiag(b, a, -c(b[1] + c[1], a[1:N - 1] + b[2:N] + c[2:N], a[N] + c[N + 1]))
shift.inv.tri(Q, mu=rep(1,dim(Q)[1]), v0_tilde=rep(1,dim(Q)[1]), zstart=6,
 digit.thresh = 6)
```

tri.sol *Solve the linear equation (-Q-zI)w=v.*

Description

Construct the solution of linear equation (-Q-zI)w=v.

Usage

```
tri.sol(Q, z, v)
```

Arguments

Q	The given tridiagonal matrix.
z	The Rayleigh shift.
v	The column vector on the right hand of equation.

Value

A solution sequence w to the equation (-Q-zI)w=v.

Examples

```
a = c(1:7)^2
b = c(1:7)^2
c = rep(0, length(a) + 1)
c[length(a) + 1] = 8^2
N = length(a)
zstart = 6
Q = tridiag(b, a, -c(b[1] + c[1], a[1:N - 1] + b[2:N] + c[2:N], a[N] + c[N + 1]))
tri.sol(Q, z=zstart, v=rep(1,dim(Q)[1]))
```

tridiag *Tridiagonal matrix*

Description

Generate tridiagonal matrix Q based on three input vectors.

Usage

```
tridiag(upper, lower, main)
```

Arguments

`upper`	The upper diagonal vector.
`lower`	The lower diagonal vector.
`main`	The main diagonal vector.

Value

A tridiagonal matrix is returned.

Examples

```
a = c(1:7)^2
b = c(1:7)^2
c = -c(1:8)^2
tridiag(b, a, c)
```

Index