Package 'expoRkit'

August 29, 2016

Type Package

Title Expokit in R

Version 0.9

Date 2012-10-08

Maintainer Niels Richard Hansen <Niels.R.Hansen@math.ku.dk>

Depends R (>= 2.14.1), methods

Imports Matrix, SparseM

Description An R-interface to the Fortran package Expokit.

URL https://bitbucket.org/nielsrhansen/exporkit/

License GPL (>= 2)

LazyLoad yes

Collate 'expoRkit.R' 'expv.R' 'expoRkit-package.R' 'data.R'

Author Roger B. Sidje [aut, cph], Niels Richard Hansen [aut, cre, cph]

Repository CRAN

Date/Publication 2012-10-08 16:53:08

NeedsCompilation yes

R topics documented:

expoRkit *Expokit in R*

Description

R-interface to the Fortran package Expokit for matrix exponentiation.

Details

Package:	expoRkit
Type:	Package
Version:	0.9
Date:	2012-10-08
License:	GPL (>= 2)
LazyLoad:	yes

Expokit is an efficient Fortran implementation for computing the matrix exponential, or rather, its action on a vector, for large sparse matrices. This can also be understood as computing the solution of a system of linear ordinary first order differential equations. This package provides an R-interface to some of the Fortran subroutines from Expokit.

The Fortran package was developed by Roger B. Sidje, see http://www.maths.uq.edu.au/expokit/. Niels Richard Hansen adapted the package for use with R and wrote the R interface. Permission to distribute the Expokit source under GPL was obtained from Roger B. Sidje.

Author(s)

Roger B. Sidje <roger.b.sidje@ua.edu>, Niels Richard Hansen <Niels.R.Hansen@math.ku.dk>.

References

Sidje, R. B. (1998) Expokit. Software Package for Computing Matrix Exponentials. ACM Trans. Math. Softw. 24(1), 130-156.

See Also

expv

expv *Matrix exponentiation and more.*

Description

Methods for computing the solution of the ODE

$$w'(t) = xw(t) + u$$

with initial condition $w(0) = v$ at one or more time points.

Usage

```
  ## S4 method for signature 'matrix,vector'
expv(x, v, t = 1, u = NULL,
    Markov = FALSE, transpose = Markov, ...)

  ## S4 method for signature 'CsparseMatrix,vector'
expv(x, v, t = 1,
    u = NULL, Markov = FALSE, transpose = Markov, ...)

  ## S4 method for signature 'dgCMatrix,vector'
expv(x, v, t = 1, u = NULL,
    Markov = FALSE, transpose = Markov, ...)

  ## S4 method for signature 'TsparseMatrix,vector'
expv(x, v, t = 1,
    u = NULL, Markov = FALSE, transpose = Markov, ...)

  ## S4 method for signature 'matrix.csc,vector'
expv(x, v, t = 1, u = NULL,
    Markov = FALSE, transpose = Markov, ...)

  ## S4 method for signature 'matrix.csr,vector'
expv(x, v, t = 1, u = NULL,
    Markov = FALSE, transpose = Markov, ...)

  ## S4 method for signature 'matrix.coo,vector'
expv(x, v, t = 1, u = NULL,
    Markov = FALSE, transpose = Markov, ...)
```

Arguments

x	a matrix.
v	a numeric vector. The initial value.
t	a numeric vector of time points at which the solution is computed.
u	a numeric vector. Default NULL.
Markov	logical. If TRUE the matrix is taken to be an rate matrix and steps are taken to ensure that the computed result is a probability vector. Default FALSE.
transpose	logical. If TRUE transpose the matrix before the solution is computed. Default equals Markov.
...	other arguments passed to Rexpv.

Details

Analytically the solution is given as

$$w(t) = \exp(tx)v + t\phi(tx)u$$

with $\phi(z) = (\exp(z) - 1)/z$. For large matrices x the computation of the full matrices $\exp(tx)$ and $\phi(tx)$ is slow. An alternative is to compute the solution w directly for a given initial value v using Krylov subspace methods. This is, in particular, efficient for large sparse matrices.

Note that if Q is a rate matrix for a homogeneous continuous time Markov process (non-negative off-diagonals and row-sums 0) and v is a probability vector (the initial distribution at time point 0), then the distribution at time point t solves

$$w'(t) = Q^T w(t).$$

In this case we want to take x to be Q^T to get the desired solution.

The solution is computed using the Fortran package Expokit. Methods are available for matrix classes implemented in the Matrix package as well as the SparseM package. The implementation avoids the computation of the full matrix exponential of tx and the approach is advantageous when we want to compute $w(t)$ for one or a few initial values v. The full matrix exponential should *not* be computed this way by looping over n different initial values.

Though there is a method implemented for ordinary (dense) matrices, such a matrix is simply coerced into a `CsparseMatrix` before the solution is computed. It is recommended that large sparse matrices are stored and handled as such, e.g. using the classes and methods from the Matrix package. Dense intermediates should be avoided.

The `x` matrix is allowed to be a dense complex matrix in which case `v` and `u` are also allowed to be complex.

Value

An n by k matrix with k the length of `t` and n the dimension of the matrix `x`. The i'th column contains the solution of the ODE at time point i.

Author(s)

Niels Richard Hansen <Niels.R.Hansen@math.ku.dk>

See Also

Rexpv, expm, expm

Examples

```
### Small 4 by 4 example.
x <- matrix(c(-1, 0, 1, 0,
              0, -2, 0, -3,
              1, 0, 0, 0,
              0, 2, -1, 0),
            4, 4)
v <- c(1, 1, 1, 1)
```

```
require(Matrix)
require(SparseM)

w <- cbind(padm(x) %*% v,
           expv(x, v),
           expv(Matrix(x, sparse = TRUE), v),
           expv(as.matrix.coo(x), v),
           expv(as.matrix.csr(x), v),
           expv(as.matrix.csc(x), v)
           )

stopifnot(all.equal(w[, 1], w[, 2]),
          all.equal(w[, 1], w[, 3]),
          all.equal(w[, 1], w[, 4]),
          all.equal(w[, 1], w[, 5]),
          all.equal(w[, 1], w[, 6]))

u <- c(2, 0, 1, 1)
ex <- padm(x)
w <- cbind(ex %*% v + (ex - diag(1, 4)) %*% solve(x, u),
           expv(x, v, u = u),
           expv(Matrix(x, sparse = TRUE), v, u = u),
           expv(as.matrix.coo(x), v, u = u),
           expv(as.matrix.csr(x), v, u = u),
           expv(as.matrix.csc(x), v, u = u)
           )

stopifnot(all.equal(w[, 1], w[, 2]),
          all.equal(w[, 1], w[, 3]),
          all.equal(w[, 1], w[, 4]),
          all.equal(w[, 1], w[, 5]),
          all.equal(w[, 1], w[, 6]))

############################################################
### Linear birth-death Markov process with immigration
############################################################

alpha <- 2.1  ## Death rate per individual
beta <- 2     ## Birth rate per individual
delta <- 20   ## Immigration rate

n <- 500L     ## state space {0, ..., n-1}
i <- seq(1, n)
rates <- c(alpha * i[-1], ## subdiagonal
           -(c(0, alpha * i[-1]) +
             c(delta, beta * i[-c(1,n)] + delta, 0)), ## diagonal
           c(delta, beta * i[-c(1, n)] + delta)) ## superdiagonal
j <- c(i[-n], i, i[-1])
i <- c(i[-1], i, i[-n])

## Sparse rate matrix constructed without dense intermediate
require(Matrix)
```

```
Q <- sparseMatrix(i = i, j = j, x = rates, dims = c(n, n))

## Evolution of uniform initial distribution
p0 <- rep(1, n)/n
time <- seq(0, 10, 0.2)
Pt <- expv(Q, p0, t = time, Markov = TRUE)

## Not run:
matplot(1:n, Pt, type = "l")
image(time, 0:(n-1), -t(Pt), col = terrain.colors(100))

## End(Not run)
```

---

orani *Australia Economic Model data, 1968-68.*

Description

This sparse matrix of order 2,529 and with 90,158 non-zero elements was used in Sidje (1998) to illustrate the application of Expokit.

Author(s)

Niels Richard Hansen <Niels.R.Hansen@math.ku.dk>

References

Sidje, R. B. (1998) Expokit. Software Package for Computing Matrix Exponentials. ACM Trans. Math. Softw. 24(1), 130-156.

See Also

expv

Examples

```
data(orani)  ## Load the data as a 'dgCMatrix' (CCS format)
v <- rep(1, 2529)
### Solving a system of 2529 coupled linear differential equations
system.time(wCCS <- expv(orani, v = v, t = 10))
oraniCOO <- as(orani, "TsparseMatrix")  ## Coerce to COO format
### In this case, the COO format gives a slight increase in
### computational time as reported in Sidje (1998).
system.time(wCOO <- expv(oraniCOO, v = v, t = 10))

print(cbind(wCCS[1:5], wCOO[1:5]), digits = 14)
```

padm *Pade approximation of dense matrix exponential.*

Description

R wrapper of the Expokit subroutines __PADM for dense matrix exponentiation via the Pade approximation.

Usage

```
padm(x, t = 1, order = 6L)
```

Arguments

x	numeric or complex matrix.
t	time. Default 1.
order	integer, the order of the Pade approximation. The default (6) is usually enough.

Details

The underlying Fortran routines compute the matrix exponential using a combination of scaling and squaring and the irreducible rational Pade approximation.

Value

The matrix exponential, $\exp(tx)$, as a numeric or complex matrix.

Author(s)

Niels Richard Hansen <Niels.R.Hansen@math.ku.dk>

References

Sidje, R. B. (1998) Expokit. Software Package for Computing Matrix Exponentials. ACM Trans. Math. Softw. 24(1), 130-156.

See Also

expm, expm, expv

Rexpv *Expokit __EXPV and __PHIV wrapper.*

Description

R wrapper of the Expokit Fortran subroutines __EXPV and __PHIV for sparse matrix exponentiation. In general, these routines compute the solution at time point t of the ODE

$$w'(t) = xw(t) + u$$

with initial condition $w(0) = v$.

Usage

```
Rexpv(a, ia, ja, n, v, t = 1, storage = "CCS", u = NULL,
  anorm = max(abs(a)), Markov = FALSE, m = 30L, tol = 0,
  itrace = 0L, mxstep = 10000L)
```

Arguments

a	numeric or complex non-zero entries in the x-matrix.
ia	integer index/pointer. Precise meaning depends on storage format.
ja	integer index/pointer. Precise meaning depends on storage format.
n	dimension of the (square) matrix.
v	numeric or complex vector.
t	time. Default 1.
storage	character, one of 'CCS' (Compressed Column Storage), 'CRS' (Compressed Row Storage) or 'COO' (COOrdinate list). Default 'CCS'.
u	numeric or complex vector. Default NULL.
anorm	A norm of the matrix. Default is the sup-norm.
Markov	logical, if TRUE the (transposed) matrix is taken to be an intensity matrix and steps are taken to ensure that the computed result is a probability vector. Default FALSE.
m	integer, the maximum size for the Krylov basis.
tol	numeric. A value of 0 (default) means square root of machine eps.
itrace	integer, 0 (default) means no trace information from Expokit, 1 means print 'happy breakdown', and 2 means all trace information printed from Expokit.
mxstep	integer. Maximum allowable number of integration steps. The value 0 means an infinite number of steps. Default 10000.

Details

The `Rexpv` function is the low level wrapper of the Fortran subroutines in the Expokit package. It is not intended to be used directly but rather via the `expv` methods. In the call the correct storage format in terms of the vectors `a`, `ia` and `ja` has to be specified via the `storage` argument. For CCS, `ia` contains 1-based row numbers of non-zero elements and `ja` contains 1-based pointers to the initial element for each column. For CRS, `ja` contains 1-based column numbers of non-zero elements and `ia` are 1-based pointers to the initial element for each row. For COO, `ia` and `ja` contain the 1-based column and row numbers, respectively, for the non-zero elements.

Value

The solution, w, of the ODE as a `numeric` or `complex` vector of length n.

Author(s)

Niels Richard Hansen <Niels.R.Hansen@math.ku.dk>

References

Sidje, R. B. (1998) Expokit. Software Package for Computing Matrix Exponentials. ACM Trans. Math. Softw. 24(1), 130-156.

See Also

`expm`, `expm`, `expv`

Examples

```
### A CCS 4 by 4 real matrix. The last element in 'ja' is the number of
### non-zero elements + 1.
a <- c(-1, 1, -2, -3, 1, 2, -1)
ia <- c(1, 3, 2, 4, 1, 2, 3)
ja <- c(1, 3, 5, 6, 8)

v <- c(1, 1, 1, 1)
wCCS <- expoRkit:::Rexpv(a, ia, ja, 4, v = v)

### COO storage instead.
ja <- c(1, 1, 2, 2, 3, 4, 4)
wCOO <- expoRkit:::Rexpv(a, ia, ja, 4, v = v, storage = 'COO')

### CRS storage instead.
a <- c(-1, 1, -2, 2, 1, -1, -3)
ja <- c(1, 3, 2, 4, 1, 4, 2)
ia <- c(1, 3, 5, 7, 8)
wCRS <- expoRkit:::Rexpv(a, ia, ja, 4, v = v, storage = 'CRS')

cbind(wCCS, wCOO, wCRS)

stopifnot(all.equal(wCCS, wCOO),
          all.equal(wCCS, wCRS),
```

```
          all.equal(wCRS, wCOO))

## Complex version
a <- c(-1, 1i, -2, -3i, 1, 2i, -1)
ia <- c(1, 3, 2, 4, 1, 2, 3)
ja <- c(1, 3, 5, 6, 8)

v <- c(1, 1, 1, 1)
wCCS <- expoRkit:::Rexpv(a, ia, ja, 4, v = v)

### COO storage instead.
ja <- c(1, 1, 2, 2, 3, 4, 4)
wCOO <- expoRkit:::Rexpv(a, ia, ja, 4, v = v, storage = 'COO')

### CRS storage instead.
a <- c(-1, 1, -2, 2i, 1i, -1, -3i)
ja <- c(1, 3, 2, 4, 1, 4, 2)
ia <- c(1, 3, 5, 7, 8)
wCRS <- expoRkit:::Rexpv(a, ia, ja, 4, v = v, storage = 'CRS')

cbind(wCCS, wCOO, wCRS)

stopifnot(all.equal(wCCS, wCOO),
          all.equal(wCCS, wCRS),
          all.equal(wCRS, wCOO))
```

Index