Package ‘multbxxc’

November 15, 2019

Type    Package
Title   Auxiliary Routines for Influx Software
Version 1.0.1
Date    2019-11-12
Author  Serguei Sokol
Maintainer Serguei Sokol <sokol@insa-toulouse.fr>

Description
Contains auxiliary routines for influx software. This package is not intended to be used directly. Influx was published here: Sokol et al. (2012) <doi:10.1093/bioinformatics/btr716>.

License GPL (>= 2)

Imports Rcpp (>= 1.0.0)

LinkingTo Rcpp, RcppArmadillo, rmumps

Depends R (>= 3.4), rmumps (>= 5.2.1-6)

URL https://metasys.insa-toulouse.fr/software/influx/

RoxygenNote 6.1.1

Encoding UTF-8

Suggests testthat

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-11-15 13:10:02 UTC

R topics documented:

bop ......................................................... 2
ij2ijv_i .................................................. 3
iv2v ...................................................... 3
jrhs_ff .................................................... 4
match_ij ................................................... 4
mm_xpf .................................................... 5
multbxxc .................................................. 5
Bloc Operation in Place

Description

src array is added (if sop=="+=") to dst[...] or any other manipulation is made according to sop parameter. Both arrays are supposed to be of type 'double'. The operation is done 'in place' without new memory allocation for dst src is reshaped and possibly replicated to fit the designated block of dst. mv can be:

- a 1 or 3 component vector describing the block: 1-margin number of dst, 2-offset, 3-length if only the margin is present than offset is 0 and length is the total length of this margin
- a matrix of indexes. Its column number must be equal to the length(dim(dst)) each row of this matrix is a multidimensional index in dst array.

sop is one of: "=" (copy src to dst[]), "+="", "+="", "*="", "/="

Usage

bop(dst, mv, sop, src)

Arguments

dst  A numeric array, destination
mv   An integer vector or matrix, describe margins to operate on
sop  A string, describes an operator to apply
src  A numeric array, source (may be replicated to fit the size of dst)

Value

None

Examples

a=matrix(1, 3, 3) # 3x3 matrix of 1's
b=1:3
bop(a, 2, "+=" b) # a += b, here b will be repeated
a
# [,1] [,2] [,3]
# [1,] 2 2 2
# [2,] 3 3 3
# [3,] 4 4 4
**ij2ijv_i**  
*Transform Repeated Matrix Indexes*

**Description**
Transforms a couple of index vectors ir and jc (ij of a sparse matrix) with possibly repeated values into sparse indexes i,j and a vector of 1d indexes of non zero values. The response can be then used for repeated creation of sparse matrices with the same pattern by calling `iv2v()` ir and jc are supposed to be sorted in increasing order, column-wise (ic runs first)

**Usage**
```
ij2ijv_i(ir, jc)
```

**Arguments**
- **ir**: An integer vector, row indexes
- **jc**: An integer vector, column indexes

**Value**
A list with fields i, j and iv

---

**iv2v**  
*Sum non Zero Repeated Values*

**Description**
sum values in v according to possibly repeated indexes in iv

**Usage**
```
iv2v(iv, v)
```

**Arguments**
- **iv**: An integer vector, obtained with `ij2ijv_i(...)$iv`
- **v**: A numeric vector

**Value**
Numeric vector
jrhs_ff  
*Update Matrix by a Cascade of Dot Product*

**Description**  
Update Matrix by a Cascade of Dot Product

**Usage**  
jrhs_ff(jrhs, ff, xpfw)

**Arguments**
- **jrhs**: A sparse matrix of type slam
- **ff**: A sparse matrix of type slam
- **xpfw**: A numeric matrix

---

match_ij  
*Fast Match for Matrix Indexes*

**Description**  
Match ix, jx-couple in ti, tj-table and return their 1-based positions (0 for non matched couples)

**Usage**  
match_ij(ix, jx, ti, tj)

**Arguments**
- **ix**: An integer vector
- **jx**: An integer vector
- **ti**: An integer vector
- **tj**: An integer vector

**Value**
An integer vector

**Examples**

```
match_ij(1:2, 1:2, 0:4, 0:4)
# [1] 2 3
```
**Dot Product SparseMatrix*DenseArray**

**Description**

Dot product of simple triplet matrix x (m x n) (measurement matrix) and a dense array y (n x k x l). Only slices of y_ from lsel vector are used.

**Usage**

```
mm_xpf(x, y_, lsel)
```

**Arguments**

- **x**: A list, sparse matrix of type slam
- **y_**: A numeric 3d array
- **lsel**: An integer vector

**Value**

An array with dimensions (m x len(lsel) x k), i.e. it is permuted on the fly.

---

**multbxxc**

**multbxxc**: Auxiliary Routines for Influx Software

**Description**

The multbxxc package provides a series C++ function most often operating inplace.

**keyword**

metabolic flux analysis (MFA)

**Author(s)**

Serguei Sokol

**References**

mult_bxxc

*Calculate Inplace a Series of Dot Product*

**Description**

Calculate Inplace a Series of Dot Product

**Usage**

`mult_bxxc(a, b, c)`

**Arguments**

- **a**: A dense array, the size of `a` is `(nr_b, nc_c, ntico)`
- **b**: A sparse matrix (cf. `simple_triplet_matrix`) of size `(nr_b*ntico, nc_b)` given by its fields `v`, `i`, and `j` describing triplet storage.
- **c**: A dense array, the size of `c` is `(ldc, nc_c, ntico)`, `ldc` must be \( \geq \) `ncol(b)`

**Value**

None

---

redim

*New Dimensions*

**Description**

Write new dimension vector while keeping the old memory

**Usage**

`redim(x, di)`

**Arguments**

- **x**: A numeric array
- **di**: An integer vector, new dimensions

**Value**

None

**Examples**

```r
a = matrix(as.double(1:12), 6, 2)
redim(a, c(3, 4))
dim(a)
# [1] 3 4
```
**resize**

*New Dimensions with Resizing*

**Description**

Write new dimension vector while keeping the old memory if possible. New memory cannot be greater than the very first allocation.

**Usage**

```r
resize(x_, di)
```

**Arguments**

- `x_` A numeric array
- `di` An integer vector, new dimensions

**Value**

None

**Examples**

```r
a = matrix(as.double(1:12), 6, 2)
resize(a, c(2, 2))
a
```

```
[,1] [,2]
[1,]  1  3
[2,]  2  4
```

---

**solve_ieu**

*Solve ODE System by Implicite Euler Scheme*

**Description**

The system is defined as \( M \frac{dx}{dt} = a \cdot x + s \) where \( M \) is a diagonal matrix given by its diagonal vector \( M \) (which has a form of matrix for term-by-term multiplication with \( x_0 \)). In discrete terms, \( (M/dt_i - a) \cdot x_i = (M/dt_i) \cdot x(i-1) + s_i \). The rmumps matrix \( (M/dt_i - a) \) is stored in list ali as XPtr<Rmumps> or a plain dense inverted matrix. Calculations are done in-place so \( s \) is modified and contains the solution on exit. The others parameters are not modified.

**Usage**

```r
solve_ieu(invdt, x0_, M, ali, s, ilua)
```
Arguments

invdt  A numeric vector, represents 1/dt
x0_   A numeric matrix or NULL, is the starting value at t0 (NULL means 0)
M     A numeric matrix representing diagonal terms (masses)
ali   A list of matrices or Rmumps objects
s     A 3d numeric array, is the source term, its last margin corresponds to time. 
     s[, , i] can be a matrix or a vector (== 1-column matrix)
ilua  An integer vector, ilua[i] gives the list index in ali for a given dt_i. In such a 
     way, ali may be shorter than time points.

Value

None
Index

bop, 2

ij2ijv_i, 3
iv2v, 3

jrhs_ff, 4

match_ij, 4
mm_xpf, 5
mult_bxxc, 6
multbxxc, 5
multbxxc-package (multbxxc), 5

redim, 6
resize, 7

simple_triplet_matrix, 6
solve_ieu, 7