Package ‘pbdBASE’

October 13, 2016

Type Package
Title Programming with Big Data -- Base Wrappers for Distributed Matrices
Version 0.4-5
Description An interface to and extensions for the 'PBLAS' and 'ScalAPACK' numerical libraries. This enables R to utilize distributed linear algebra for codes written in the 'SPMD' fashion. This interface is deliberately low-level and mimics the style of the native libraries it wraps. For a much higher level way of managing distributed matrices, see the 'pbdDMAT' package.
License Mozilla Public License 2.0
Depends R (>= 3.0.0), methods
Imports pbdMPI (>= 0.3-1), pbdSLAP(>= 0.2-1), utils
SystemRequirements OpenMPI (>= 1.5.4) on Solaris, Linux, Mac, and FreeBSD. MS-MPI (Microsoft HPC Pack 2012) or MPICH2 (>= 1.4.1p1) on Windows.
LazyLoad yes
LazyData yes
ByteCompile yes
NeedsCompilation yes
URL http://r-pbd.org/
BugReports http://group.r-pbd.org/
MailingList Please send questions and comments regarding pbdR to RBigData@gmail.com
Maintainer Drew Schmidt <schmidt@math.utk.edu>
RoxygenNote 5.0.1
Author Drew Schmidt [aut, cre], Wei-Chen Chen [aut], George Ostrouchov [aut], Pragneshkumar Patel [aut], Ewan Higgs [ctb]
**R topics documented:**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>pbdBASE-package</td>
<td>3</td>
</tr>
<tr>
<td>BASE Global Environment</td>
<td>4</td>
</tr>
<tr>
<td>base.crossprod</td>
<td>4</td>
</tr>
<tr>
<td>base.dallreduce</td>
<td>5</td>
</tr>
<tr>
<td>base.descinit</td>
<td>5</td>
</tr>
<tr>
<td>base.dgamx2d</td>
<td>6</td>
</tr>
<tr>
<td>base.dgesd2d</td>
<td>6</td>
</tr>
<tr>
<td>base.dhilbmk</td>
<td>7</td>
</tr>
<tr>
<td>base.dim0</td>
<td>8</td>
</tr>
<tr>
<td>base.igamx2d</td>
<td>8</td>
</tr>
<tr>
<td>base.igsum2d</td>
<td>9</td>
</tr>
<tr>
<td>base.indxg2p</td>
<td>9</td>
</tr>
<tr>
<td>base.matexp</td>
<td>10</td>
</tr>
<tr>
<td>base.maxdim</td>
<td>10</td>
</tr>
<tr>
<td>base.minctxt</td>
<td>11</td>
</tr>
<tr>
<td>base.mksubmat</td>
<td>11</td>
</tr>
<tr>
<td>base.nbd</td>
<td>12</td>
</tr>
<tr>
<td>base.numroc</td>
<td>12</td>
</tr>
<tr>
<td>base.ownany</td>
<td>13</td>
</tr>
<tr>
<td>base.pdchtri</td>
<td>14</td>
</tr>
<tr>
<td>base.pdclvar</td>
<td>14</td>
</tr>
<tr>
<td>base.pdhilbmk</td>
<td>15</td>
</tr>
<tr>
<td>base.pdmkcpn1</td>
<td>15</td>
</tr>
<tr>
<td>base.pdmvsum</td>
<td>16</td>
</tr>
<tr>
<td>base.pdsweep</td>
<td>16</td>
</tr>
<tr>
<td>base.procgird</td>
<td>17</td>
</tr>
<tr>
<td>base.p_matexp_pade_wrap</td>
<td>17</td>
</tr>
<tr>
<td>base.p_matpow_by_squaring_wrap</td>
<td>18</td>
</tr>
<tr>
<td>base.rcolcpy</td>
<td>18</td>
</tr>
<tr>
<td>base.rcolcpy2</td>
<td>19</td>
</tr>
<tr>
<td>base.redist</td>
<td>19</td>
</tr>
<tr>
<td>base.rl2blas</td>
<td>20</td>
</tr>
<tr>
<td>base.rl2insert</td>
<td>20</td>
</tr>
<tr>
<td>base.rpdgecon</td>
<td>21</td>
</tr>
<tr>
<td>base.rpdgecon</td>
<td>21</td>
</tr>
<tr>
<td>base.rpdgels</td>
<td>22</td>
</tr>
<tr>
<td>base.rpdgemm</td>
<td>22</td>
</tr>
<tr>
<td>base.rpdgemr2d</td>
<td>23</td>
</tr>
<tr>
<td>base.rpdgeqpf</td>
<td>23</td>
</tr>
<tr>
<td>base.rpdgesv</td>
<td>23</td>
</tr>
<tr>
<td>base.rpdgesvd</td>
<td>24</td>
</tr>
<tr>
<td>base.rpdgetrf</td>
<td>24</td>
</tr>
<tr>
<td>base.rpdgetri</td>
<td>25</td>
</tr>
</tbody>
</table>
pbdbaseMpackage

ScaLAPACK Wrappers and Utilities

Description
A package contains the basic methods for dealing with distributed data types, as well as the data types themselves.

Details

Package: pbdbase
Type: Package
License: MPL
LazyLoad: yes

This package requires an MPI library (OpenMPI, MPICH2, or LAM/MPI).
Author(s)

Drew Schmidt <schmidt AT math.utk.edu>, Wei-Chen Chen, George Ostrouchov, and Pragneshkumar Patel.

References

Programming with Big Data in R Website: http://r-pbd.org/

---

**Global Environment for the pbdBASE Package**

**Description**

The environment for the pbdBASE package where "global" variables are stored.

**Usage**

.base.crossprod

**Format**

An object of class environment of length 0.

**Details**

The .__blacs_gridinfo_ and .__blacs_initialized objects are stored in this environment.

---

**crossprod**

**Description**

Crossproduct.

**Usage**

base.crossprod(uplo, trans, x, descx, descc)

**Arguments**

- **uplo**: Triangle whose values to use.
- **trans**: tcrossprod or crossprod.
- **x**: Matrix to crossprod.
- **descx**: ScaLAPACK descriptor array.
- **descx**: ScaLAPACK descriptor array of output.
**base.dallreduce**

**Details**
For advanced users only.

**Description**
Allreduce

**Usage**
```
base.dallreduce(x, descx, op = "sum", scope = "All")
```

**Arguments**
- `x`: Matrix.
- `descx`: ScaLAPACK descriptor array.
- `op`: Operation.
- `scope`: Rows, columns, or both.

**Details**
For advanced users only.

---

**base.descinit**

**Description**
Creates ScaLAPACK descriptor array.

**Usage**
```
base.descinit(dim, bldim, ldim, ICTXT = 0)
```

**Arguments**
- `dim`: Global dim.
- `bldim`: Blocking dim.
- `ldim`: Local dim.
- `ICTXT`: BLACS context.

**Details**
For advanced users only.
**base.dgmx2d**

**Description**

Min value across a process grid.

**Usage**

```
base.dgmx2d(ictxtL, scopeL, mL, nL, xL, ldaL, rdestL, cdestL)
base.igamn2d(ictxtL, scopeL, mL, nL, xL, ldaL, rdestL, cdestL)
base.dgmn2d(ictxtL, scopeL, mL, nL, xL, ldaL, rdestL, cdestL)
```

**Arguments**

- **ictxt**
  - BLACS ICTXT.
- **scope**
  - Rows, cols, or both.
- **m, n**
  - Problem size.
- **x**
  - Local values.
- **lda**
  - Leading dimension.
- **rdest**
  - Row destination.
- **cdest**
  - Col destination.

**Details**

For advanced users only.

---

**base.dgesd2d**

**Description**

Sent value across a process grid.

**Usage**

```
base.dgesd2d(ictxtL, scopeL, mL, nL, xL, ldaL, rdestL, cdestL)
base.dgerv2d(ictxtL, scopeL, mL, nL, xL, ldaL, rdestL, cdestL)
```
**Arguments**

- ICTXT: BLACS ICTXT.
- SCOPE: Rows, cols, or both.
- m, n: Problem size.
- x: Local values.
- lda: Leading dimension.
- RDEST: Row destination.
- CDEST: Col destination.

**Details**

For advanced users only.

---

**Description**

Create Hilbert matrix.

**Usage**

```
base.dhilmk(n)
```

**Arguments**

- n: Size.

**Details**

For advanced users only.
**Description**

Compute dimensions on process MYROW=MYCOL=0

**Usage**

```plaintext
base.dim0(dim, ICTXT = 0)
```

**Arguments**

- `dim`: Global dim.
- `ICTXT`: BLACS context.

**Details**

For advanced users only.

---

**Description**

Max value across a process grid.

**Usage**

```plaintext
base.igamx2d(ICTXT, SCOPE, m, n, x, lda, RDEST, CDEST)
```

**Arguments**

- `ICTXT`: BLACS ICTXT.
- `SCOPE`: Rows, cols, or both.
- `m, n`: Problem size.
- `x`: Local values.
- `lda`: Leading dimension.
- `RDEST`: Row destination.
- `CDEST`: Col destination.

**Details**

For advanced users only.
**base.igsum2d**

**BLACS Sums**

**Description**
Sum across a process grid.

**Usage**

base.igsum2d(ictxtL, scopeL, mL, nL, xL, ldaL, rdestL, cdestL)

base.dgsum2d(ictxtL, scopeL, mL, nL, xL, ldaL, rdestL, cdestL)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ictxt</td>
<td>BLACS ICTXT.</td>
</tr>
<tr>
<td>scope</td>
<td>Rows, cols, or both.</td>
</tr>
<tr>
<td>mL, n</td>
<td>Problem size.</td>
</tr>
<tr>
<td>x</td>
<td>Local values.</td>
</tr>
<tr>
<td>lda</td>
<td>Leading dimension.</td>
</tr>
<tr>
<td>rdest</td>
<td>Row destination.</td>
</tr>
<tr>
<td>cdest</td>
<td>Col destination.</td>
</tr>
</tbody>
</table>

**Details**
For advanced users only.

**base.indxg2p**

**indx2p**

**Description**
Computes the process coordinate which contains the entry of a distributed matrix specified by a global index INDXGLOB. Simplified reimplementation of the ScaLAPACK aux INDXG2P function.

**Usage**

base.indxg2p(indxglobL, nbL, nprocsI)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>indxglob</td>
<td>Global index.</td>
</tr>
<tr>
<td>nb</td>
<td>Block size.</td>
</tr>
<tr>
<td>nprocs</td>
<td>Total number of processors over which matrix is distributed.</td>
</tr>
</tbody>
</table>
**base.matexp**

**Description**

Serial matrix exponentiation.

**Usage**

base.matexp(A, p = 6, t = 1)

**Arguments**

- **A**: Matrix to exponentiate.
- **p**: Pade’ expansion size.
- **t**: Scaling factor.

**Details**

For advanced users only.

---

**base.maxdim**

**Description**

Compute maximum dimension across all nodes

**Usage**

base.maxdim(dim)

**Arguments**

- **dim**: Global dim.

**Details**

For advanced users only.
**base.minctxt  Get BLACS Context Grid Information**

**Description**
Finds the smallest integers for creating a new BLACS context.

**Usage**
\[
\text{base.minctxt}(\text{after} = 0)
\]

**Arguments**
- **after** ignores all values below this integer as possibilities

**Details**
For advanced users only.
- Returns the smallest integer which could become a new BLACS context value.
- For example, if contexts 0, 1, and 2 are taken, and \(\text{after}=0\), then the function returns 3. If 0, 1, 2, and 5 are taken, the function returns 3 if \(\text{after}=0\), but returns 6 if \(\text{after}=4\).
- The function is useful when a transitory grid is needed, such as for reading in data onto a subset of processors before distributing out to the full grid.

**Value**
- Returns the minimum value.

---

**base.mksubmat  (Un)Distribute**

**Description**
(Un)Distribute matrix.

**Usage**
\[
\text{base.mksubmat}(x, \text{descx})
\]
\[
\text{base.mkgblmat}(x, \text{descx}, \text{rsrc}, \text{csrc})
\]

**Arguments**
- **x** Matrix.
- **descx** ScaLAPACK descriptor array.
- **rsrc, csrc** Row/column source.
**base.nbd**

*Next Best Divisor*

**Description**

Given integers n and d, with n>d, this function finds the "next best divisor" of n which is greater than or equal to d.

**Usage**

base.nbd(n, d)

**Arguments**

- **n**: The divident (number divided into).
- **d**: The candidate divisor.

**Details**

Suprisingly useful for thinking about processor grid shapes.

**Examples**

```r
## Not run:
library(pbdBASE, quiet = TRUE)
base.nbd(100, 10) # 10 divides 100, so 10 is returned
base.nbd(100, 11) # 11 does not, so the "next best" divisor, 20, is returned

## End(Not run)
```

---

**base.numroc**

*numroc*

**Description**

NUMber of Rows Or Columns

**Usage**

base.numroc(dim, bldim, ICTXT = 0, fixme = TRUE)
base.ownany

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim</td>
<td>Global dim.</td>
</tr>
<tr>
<td>bldim</td>
<td>Blocking dim.</td>
</tr>
<tr>
<td>ICTXT</td>
<td>BLACS context.</td>
</tr>
<tr>
<td>fixme</td>
<td>Should ldims be &quot;rounded&quot; to 0 or not.</td>
</tr>
</tbody>
</table>

Details

For advanced users only.

base.ownany

Determining Local Ownership of a Distributed Matrix

Description

For advanced users only.

Usage

```
base.ownany(dim, bldim, ICTXT = 0)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim</td>
<td>global dimension</td>
</tr>
<tr>
<td>bldim</td>
<td>blocking dimension</td>
</tr>
<tr>
<td>ICTXT</td>
<td>BLACS context</td>
</tr>
</tbody>
</table>

Details

A simple wrapper of numroc. The return is the answer to the question 'do I own any of the global matrix?'. Passing a distributed matrix is allowed, but often it is convenient to determine that information without even having a distributed matrix on hand. In this case, explicitly passing the appropriate information to the arguments `dim=`, `bldim=` (and `ctxt=` as necessary, since it defaults to 0) while leaving `x` missing will produce the desired result. See the examples below for more clarity.

The return for each function is local.

Examples

```
# Not run:
# Save code in a file "demo.r" and run with 2 processors by
# > mpiexec -np 2 Rscript demo.r

library(pbdBASE, quiet = TRUE)
init.grid()

iown <- ownany(dim=c(4, 4), bldim=c(2, 2), ICTXT=0)
```
Description
Inverse of cholesky.

Usage
base.pdchtri(uplo, x, descx, descc)

Arguments
uplo Triangle whose values to use.
x Matrix to crossprod.
descx ScaLAPACK descriptor array.
descc ScaLAPACK descriptor array of output.

Details
For advanced users only.

Description
Computes the variances of a ScaLAPACK-like distributed matrix. Significantly faster than using apply(), even in compared to the performance differences you would find comparing these two approaches using just base R.

Usage
base.pdclvar(x, descx)

Arguments
x The matrix.
descx ScaLAPACK descriptor array.
base.pdhilbmk

Description
Create Hilbert matrix.

Usage
base.pdhilbmk(descx)

Arguments
descx ScaLAPACK descriptor matrix.

Details
For advanced users only.

base.pdmkcpn1

Description
Create Companion Matrix

Usage
base.pdmkcpn1(coef, descx)

Arguments
coef Coefficients vector.
descx ScaLAPACK descriptor array.

Details
For advanced users only.
base.pdmvsum           *R-like Matrix-Vector Sum*

**Description**

For advanced users only.

**Usage**

```r
base.pdmvsum(x, descx, y, descy)
```

**Arguments**

- `x`: Matrix.
- `descx, descy`: ScaLAPACK descriptor array.
- `y`: Vector.

base.pdsweep           *pdsweep*

**Description**

Matrix-Vector Sweep

**Usage**

```r
base.pdsweep(x, descx, vec, MARGIN, FUN)
```

**Arguments**

- `x`: Matrix.
- `descx`: ScaLAPACK descriptor array.
- `vec`: Vector
- `MARGIN`: Rows or columns.
- `FUN`: Function.

**Details**

For advanced users only.
**base.procgrid**

**procgrid**

---

**Description**

"Optimal" process grid when nprow and npcol are empty

**Usage**

```
base.procgrid(nprocs)
```

**Arguments**

- `nprocs`: Number of processors.

**Details**

For advanced users only.

---

**base.p_matexp_pade_wrap**

**p_matexp_pade_wrap**

---

**Description**

Pade’ expansion.

**Usage**

```
base.p_matexp_pade_wrap(A, desca, p = 6)
```

**Arguments**

- `A`: Matrix.
- `desca`: ScaLAPACK descriptor array.
- `p`: Order of the Pade’ approximation.

**Details**

For advanced users only.
**Description**

Matrix power by squaring.

**Usage**

```r
base.p_matpow_by_squaring_wrap(A, desca, b = 1)
```

**Arguments**

- `A`: Matrix.
- `desca`: ScaLAPACK descriptor array.
- `b`: Power.

**Details**

For advanced users only.

---

**Description**

For advanced users only.

**Usage**

```r
base.rcolcpy(x, descx, y, descy, xcol, ycol)
```

**Arguments**

- `x, y`: Matrix.
- `descx, descy`: ScaLAPACK descriptor array.
- `xcol, ycol`: Columns.
**Description**

For advanced users only.

**Usage**

```r
cbase.rcolcpy2(x, descx, y, descy, xcol, ycol)
```

**Arguments**

- `x, y` Matrix.
- `descx, descy` ScaLAPACK descriptor array.
- `xcol, ycol` Columns.

---

**Description**

Redistribute a matrix from rank 0 to all ranks in block cyclic fashion.

**Usage**

```r
cbase.redist(desc, A)
```

**Arguments**

- `desc` ScaLAPACK descriptor array.
- `A` Matrix.
### base.r2blas

**Level 2 R-like BLAS**

**Description**

For advanced users only.

**Usage**

```r
base.r2blas(xL, descxL, vecL, funI)
```

**Arguments**

- **x**: Matrix.
- **descx**: ScaLAPACK descriptor array.
- **vec**: Global vector.
- **FUN**: Function.

### base.r2insert

**R-like Matrix-Vector Insertion**

**Description**

For advanced users only.

**Usage**

```r
base.r2insert(x, descx, vec, i, j)
```

**Arguments**

- **x**: Matrix.
- **descx**: ScaLAPACK descriptor array.
- **vec**: Global vector.
- **i, j**: Indices.
**Description**

Inverse condition number of a general matrix.

**Usage**

```plaintext
base.rpdgecon(norm, m, n, a, desca)
```

**Arguments**

- `norm`: Type of norm.
- `m, n`: Problem size
- `a`: Matrix.
- `desca`: ScaLAPACK descriptor array.

**Details**

For advanced users only.

**Description**

Linear model fitter via rank-revealing QR (with pivoting).

**Usage**

```plaintext
base.rpdgels(tol, m, n, nrhs, a, desca, b, descb)
```

**Arguments**

- `tol`: Numerical tolerance for the QR.
- `m, n`: Problem size.
- `nrhs`: Number of right hand sides.
- `a`: Left hand side.
- `desca`: ScaLAPACK descriptor array.
- `b`: Right hand side.
- `descb`: ScaLAPACK descriptor array.

**Details**

For advanced users only.
**Description**

Matrix-Matrix Multiply.

**Usage**

base.rpdgemm(transxL, transyL, xL, descxL, yL, descyL, descL)

**Arguments**

- `transxL, transy`  'T' or 'N' for transpose or not.
- `xL, yL`  Matrix.
- `descxL, descyL, descL`  ScaLAPACK descriptor array.

**Details**

For advanced users only.

---

**Description**

General 2d block cyclic redistribution function.

**Usage**

base.rpdgmr2d(x, descx, descy)

**Arguments**

- `x`  Matrix.
- `descx, descy`  ScaLAPACK descriptor array.

**Details**

For advanced users only.
**base.rpdgeqpf**

*Description*
QR.

*Usage*
base.rpdgeqpf(tol, m, n, x, descx)

*Arguments*
tol Numerical tolerance for the QR.
m, n Problem size.
x Matrix.
descx ScaLAPACK descriptor array.

*Details*
For advanced users only.

---

**base.rpdgesv**

*Description*
Solving a (square) system of equations.

*Usage*
base.rpdgesv(n, nrhs, a, desca, b, descb)

*Arguments*
n Problem size.
nrhs Number of right hand sides.
a, b Matrix.
desca, descb ScaLAPACK descriptor array.

*Details*
For advanced users only.
Description
SVD.

Usage
base.rpdgesvd(jobuL, jobvtL, mL, nL, aL, descaL, descuL, descvtL, ..., inplace = false)

Arguments
jobu, jobvt Control for u/vt return.
m, n Problem size.
a Matrix.
desca, descu, descvt ScaLAPACK descriptor array.
... Ignored
inplace Should the computation be done in-place or not. For REALLY advanced users only.

Details
For advanced users only.

Description
LU factorization.

Usage
base.rpdgetrf(aL, desca)

Arguments
a Matrix.
desca ScaLAPACK descriptor array.

Details
For advanced users only.
**base.rpdgetri**

---

**Description**

Matrix inversion.

**Usage**

```
base.rpdgetri(n, a, desca)
```

**Arguments**

- `n` : Problem size.
- `a` : Matrix.
- `desca` : ScaLAPACK descriptor array.

**Details**

For advanced users only.

---

**base.rpdldange**

---

**Description**

Matrix norms.

**Usage**

```
base.rpdldange(norm, m, n, a, desca)
```

**Arguments**

- `norm` : Type of norm.
- `m, n` : Problem size
- `a` : Matrix.
- `desca` : ScaLAPACK descriptor array.

**Details**

For advanced users only.
**base.rpdlaprnt**  
*rpdlaprint*

---

**Description**
Matrix printer.

**Usage**
```plaintext
base.rpdlaprnt(m, n, a, desca)
```

**Arguments**
- **m, n**: Number rows/cols.
- **a**: Matrix.
- **desca**: ScaLAPACK descriptor array.

**Details**
For advanced users only.

---

**base.rpdorgqr**  
*rpdorgqr*

---

**Description**
Recover Q.

**Usage**
```plaintext
base.rpdorgqr(m, n, k, qr, descqr, tau)
```

**Arguments**
- **m, n**: Problem size.
- **k**: Number of elementary reflectors.
- **qr**: QR decomposition.
- **descqr**: ScaLAPACK descriptor array.
- **tau**: Elementary reflectors.

**Details**
For advanced users only.
### Description

\( \text{op}(Q) \ast y \).

### Usage

```c
base.rpdormqr(side, trans, m, n, k, qr, descqr, tau, c, descc)
```

### Arguments

- **side**: 'L' or 'R', for left or right application of Q matrix.
- **trans**: Q or Q^T.
- **m, n**: Problem size.
- **k**: Number of elementary reflectors.
- **qr**: QR decomposition.
- **descqr**: ScaLAPACK descriptor array.
- **tau**: Elementary reflectors.
- **c**: Vector.
- **descc**: ScaLAPACK descriptor array.

### Details

For advanced users only.

### Description

Cholesky factorization.

### Usage

```c
base.rpdpotrf(uplo, n, a, desca)
```

### Arguments

- **uplo**: Triangle where the information is stored (in the symmetric matrix).
- **n**: Problem size.
- **a**: Matrix.
- **desca**: ScaLAPACK descriptor array.
Details

For advanced users only.

---

**base.rpdsyevr**  
**rpdsyevr**

**Description**

Symmetric eigenvalue decomposition.

**Usage**

```
base.rpdsyevr(jobz, uplo, n, a, desca, descz)
```

**Arguments**

- **jobz**: Control for if vectors/values/both are returned.
- **uplo**: Triangle where the information is stored (in the symmetric matrix).
- **n**: Problem size.
- **a**: Matrix.
- **desca, descz**: ScaLAPACK descriptor array.

Details

For advanced users only.

---

**base.rpdsyevx**  
**rpdsyevx**

**Description**

Gereneralized eigenvalue problem.

**Usage**

```
base.rpdsyevx(jobz, range, n, a, desca, vl, vu, il, iu, abstol = 1e-08, orfac = 0.001)
```
Arguments

jobz  
Control for if vectors/values/both are returned.

range  
Parameter to determine the search criteria for eigenvalues.

n  
Problem size.

a  
Matrix.

desca  
ScaLAPACK descriptor array.

vl, vu  
Endpoints of the interval subset of the real line in which to search for eigenvalues, if specified by range.

il, iu  
Eigenvalues with indices il, ..., iu will be found, if specified by range.

abstol  
Absolute error tolerance for the eigenvalues.

orfac  
Eigenvectors with eigenvalues below orfac*norm(a) of each other are reorthogonalized.

Details

For advanced users only.

Description

Transpose.

Usage

base.rpdtran(a, desca, descc)

Arguments

a  
Matrix.

desca, descc  
ScaLAPACK descriptor array.

Details

For advanced users only.
**Description**

Inverse condition number of a triangular matrix.

**Usage**

```r
dx = base.rpdtrcon(norm, uplo, diag, n, a, desca)
```

**Arguments**

- `norm`: Type of norm.
- `uplo`: Triangle where information is stored.
- `diag`: Specifies if the matrix is unit triangular or not.
- `n`: Problem size
- `a`: Matrix.
- `desca`: ScaLAPACK descriptor array.

**Details**

For advanced users only.

---

**Description**

For advanced users only.

**Usage**

```r
dx = base.rrowcpx(x, descx, y, descy, xrow, yrow)
```

**Arguments**

- `x, y`: Matrix.
- `descx, descy`: ScaLAPACK descriptor array.
- `xrow, yrow`: Rows.
**base.rrowcpy2**  
*R Row Copy-2*

**Description**
For advanced users only.

**Usage**
```
base.rrowcpy2(x, descx, y, descy, xrow, yrow)
```

**Arguments**
- `x, y`  
  Matrix.
- `descx, descy`  
  ScaLAPACK descriptor array.
- `xrow, yrow`  
  Rows.

---

**base.tri2zero**  
*tri2zero*

**Description**
Zero Triangle

**Usage**
```
base.tri2zero(x, descx, uplo = "L", diag = "N")
```

**Arguments**
- `x`  
  Matrix.
- `descx`  
  ScaLAPACK descriptor array.
- `uplo`  
  Triangle.
- `diag`  
  Zero diagonal as well.

**Details**
For advanced users only.
### base.valid_context  
**BLACS Context Validation**

**Description**
Checks if a supplied ICTXT is valid.

**Usage**
```
base.valid_context(ICTXT, ..., override = FALSE)
```

**Arguments**
- **ICTXT**: BLACS context number.
- **...**: Not used.
- **override**: If override=FALSE, the context number will produce an error if it is any of the reserved contexts (0, 1, or 2).

### blacsexit  
**BLACS Exit**

**Description**
Shuts down all BLACS communicators.

**Usage**
```
base.blacsexit(CONT = TRUE)

blacsexit(CONT = TRUE)
```

**Arguments**
- **CONT**: logical; determines whether or not to shut down all MPI communicators

**Details**
If the user wishes to shut down BLACS communicators but still have access to MPI, then call this function with CONT=TRUE. Calling blacsexit(CONT=FALSE) will shut down all MPI communicators, equivalent to calling
```
> blacsexit(CONT=TRUE) > finalize(mpi.finalize=TRUE)
```
This function is automatically invoked if BLACS communicators are running and finalize() is called.
Value

Has an invisible return of 0 when successful.

Examples

```r
## Not run:
# Save code in a file "demo.r" and run with 2 processors by
# > mpiexec -np 2 Rscript demo.r

library(pbdBASE, quiet = TRUE)
init.grid()
blacsexit()
finalize()

## End(Not run)
```

#### coords

**Local to Global/Global to Local Indexing**

Description

Get the local index given global information.

Usage

```r
indxg2l(INDXGLOB, NB, IPROC, ISRCPROC, NPROCS)
indx12g(INDXLOC, NB, IPROC, ISRCPROC, NPROCS)
```

Arguments

- **INDXGLOB**: Global index.
- **NB**: Block size.
- **IPROC**: Coordinate of the process whose local info is to be determined.
- **ISRCPROC**: The coordinate of the process that possesses the first row/column of the distributed matrix. That’s always 0 pbdDMAT.
- **NPROCS**: Total number of processors over which matrix is distributed.
- **INDXLOC**: Local index.

Details

For advanced users only.
Description

Get the local index-pair given global information.

Usage

g2lpair(gi, gj, bldim, ICTXT)

l2gpair(i, j, bldim, ICTXT)

Arguments

- gi, gj: Global indices.
- bldim: Blocking dimension
- ICTXT: BLACS context.
- i, j: Local indices.

Details

For advanced users only.

Description

Grab diagonal or create distributed diagonal matrix.

Usage

base.ddiagtk(x, descx, proc.dest = "all")

base.ddiagmk(diag, descx)

Arguments

- x: Matrix.
- descx: ScaLAPACK descriptor array.
- proc.dest: Who owns the result.
- diag: Diagonal.

Details

For advanced users only.
**finalizer**

**Finalizer**

**Description**

A replacement for `pbdMPI::finalize()` that automatically shuts BLACS communicators down.

**Usage**

```r
finalizer(mpi.finalize = Npbd_env$SPMD.CT$mpi.finalize)
```

**Arguments**

- `mpi.finalize` If MPI should be shut down.

---

**g2lcoord**

**g2lcoord**

**Description**

Global to local coordinates with explicit ownership given.

**Usage**

```r
g2lcoord(dim, bldim, gi, gj, gridinfo)
```

**Arguments**

- `dim` Global dimension.
- `bldim` Blocking dimension.
- `gi, gj` Global row and column indices, respectively.
- `gridinfo` The return of `base.blacs(ICTXT(x))`. See the Details section for more information.

**Value**

For the process that owns the desired local data at global indices `(gi, gj)`, the return is the local index. Otherwise, `NA` is returned.
Description
Global to local coords.

Usage
base.g2l_coord(ind, dim, bldim, ICTXT = 0)
g2l_coord(ind, dim, bldim, ICTXT = 0)

Arguments
ind Matrix indices.
dim Global dim.
bldim Blocking dimension.
ICTXT BLACS context.

Details
For advanced users only.

Description
Frees a BLACS context.

Usage
base.gridexit(ICTXT, override = FALSE)
ggridexit(ICTXT, override = FALSE)

Arguments
ICTXT BLACS context number.
override logical; if TRUE, ignores normal check preventing the closing of ICTXT values of 0, 1, and 2. This could cause things to go crazy and I do not recommend it.
Details

For advanced users only.

The function frees the requested BLACS context. It is a trivial wrapper for the BLACS routine BLACS_GRIDEXIT. Also removes the object .__blacs_gridinfo__ICTXT.

Contexts 0, 1, and 2 cannot be freed in this way unless the argument override=FALSE. This will probably break something and I do not recommend it.

Value

Silently returns 0 when successful. Silently returns 1 when requested ICTXT does not exist.

gridinfo

Get BLACS Context Grid Information

Description

Grabs the existing BLACS context grid information.

Usage

base.blacs(ICTXT = 0)

blacs(ICTXT = 0)

Arguments

ICTXT BLACS context number.

Details

BLACS contexts have important internal use, and advanced users familiar with ScaLAPACK might find some advantage in directly manipulating these process grids. Most users should not need to directly manage BLACS contexts, in this function or elsewhere.

The function effectively serves as a shorthand for

eval(parse(text=paste(".\_blacs\_gridinfo\_", ICTXT, sep="")))

Value

Returns a list with 5 elements: NPROW and NPCOL, the number of process rows and columns respectively; ICTXT, the associated BLACS context number; MYROW and MYCOL, the current process' row and column position in the process grid.
Examples

```r
## Not run:
# Save code in a file "demo.r" and run with 2 processors by
# > mpiexec -np 2 Rscript demo.r

library(pbdBASE, quiet = TRUE)
init.grid()

mygrid <- blacs(0)

pbdMPI::comm.print(mygrid)

finalize()

## End(Not run)
```

---

**gridinit**

**blacs_gridinit**

---

**Description**

BLACS grid initialization.

**Usage**

```r
base.blacs_gridinit(ICTXT, NPROW, NPCOL, ..., quiet = FALSE)

blacs_gridinit(ICTXT, NPROW, NPCOL, ..., quiet = FALSE)
```

**Arguments**

- **ICTXT**: BLACS context.
- **NPROW, NPCOL**: Number of process rows/cols.
- **...**: Additional arguments.
- **quiet**: Verbose initialization or not.

**Details**

For advanced users only.
**Description**

Manages the creation of BLACS context grids.

**Usage**

```plaintext
init.grid(NPROW, NPCOL, ICTXT, quiet = FALSE)
```

**Arguments**

- **NPROW**: number of process rows. Can be missing; see details.
- **NPCOL**: number of process columns. Can be missing; see details.
- **ICTXT**: BLACS context number.
- **quiet**: logical; controls whether or not information about grid size should be printed.

**Details**

`blacs_gridinit()` is for experienced users only. It is a shallow wrapper of the BLACS routine `BLACS_GRIDINIT`, with the addition of creating the `__blacs_gridinfo_ictxt` objects, as described below.

The remainder of this section applies only to `init.grid()`.

If `ICTXT` is missing, three variables will be created in the `.pdbBASEEnv` environment:

- `__blacs_gridinfo_0`
- `__blacs_gridinfo_1`
- `__blacs_gridinfo_2`

These variables store the BLACS process grid information for the BLACS context corresponding to the trailing digit of the variable. Most users should invoke `init.grid()` in this fashion, namely with `ICTXT` missing, and only do so once.

Contexts 0, 1, and 2 are reserved. Additional custom contexts are possible to create, but they must be integers $\geq 3$.

Context 0 is the “full” process grid of `NPROW` by `NPCOL` processes; contexts 1 is the process grid consisting of 1 process row and `NPROW*NPCOL` processes columns; context 2 is the process grid consisting of `NPROW*NPCOL` processes rows and 1 process column. These contexts can be redundant depending on the number of processes available.

BLACS contexts have important internal use, and advanced users familiar with ScaLAPACK might find some advantage in directly manipulating these process grids. Most users should not need to directly manage BLACS contexts, in this function or elsewhere.

If the `NPROW` and `NPCOL` values are missing, then a best process grid will be chosen for the user based on the total available number of processes. Here “best” means as close to a square grid as possible.
The variables `__blacs_gridinfo_ictxt` are just storage mechanisms to avoid needing to directly invoke the BLACS routine `BLACS_GRIDINFO`.

Additionally, another variable is created in the `.pbdbaseEnv` environment, namely `__blacs_initialized`. Its existence is to alert `finalize()` to shut down BLACS communicators, if necessary, to prevent memory leaks.

**Value**

Silently returns 0 when successful. Additionally, several variables are created in the `.pbdbaseEnv` environment. See Details section.

**Examples**

```r
## Not run:
# Save code in a file "demo.r" and run with 2 processors by
# > mpiexec -n 2 Rscript demo.r
library(pbdbase, quiet = TRUE)
init.grid()
finalize()
## End(Not run)
```

---

**l2g Coord**

**Description**

Local to global coords.

**Usage**

```r
base.l2g_coord(ind, dim, bldim, ICTXT = 0)
```

```r
l2g_coord(ind, dim, bldim, ICTXT = 0)
```

**Arguments**

- `ind`: Matrix indices.
- `dim`: Global dim.
- `bldim`: Blocking dimension.
- `ICTXT`: BLACS context.

**Details**

For advanced users only.
numroc2

---

numroc2

---

**Description**

A better version of NUMROC (NUMber Rows Or Columns). Returns the local dimension given global matrix + distribution parameters.

**Usage**

numroc2(N, NB, IPROC, NPROCS)

**Arguments**

- `N`: Global number of rows/cols.
- `NB`: Block size.
- `IPROC`: Coordinate of the process whose local info is to be determined.
- `NPROCS`: Total number of processors over which matrix is distributed.

**Details**

For advanced users only.

---

**pcoords**

*Interchange Between Process Number and BLACS Coordinates*

---

**Description**

Grabs the existing BLACS context grid information.

**Usage**

base.pnum(ICTXT, PROW, PCOL)

base.pcoord(ICTXT, PNUM)

**Arguments**

- `ICTXT`: BLACS context number.
- `PROW`, `PCOL`: BLACS grid location row/column
- `PNUM`: process rank
Details

For advanced users only. These functions are simple recreations of the BLACS routines BLACS_PNUM and BLACS_PCOORD. The former gets the process number associated with the BLACS process grid location c(MYPROW, MYPOL), while the latter does the reverse.

Value

pnum returns an integer; pcoord returns a list containing elements PROW and PCOL.

Examples

```r
## Not run:
# Save code in a file "demo.r" and run with 2 processors by
# > mpiexec -np 2 Rscript demo.r

library(pbdBASE, quiet = TRUE)
init.grid()

blacs_ <- blacs(ICTXT = 0)

# get the ICTXT = 0 BLACS coordinates for process 0
myCoords <- pcoord(ICTXT = 0, PNUM = 0)

comm.print(myCoords)

finalize()

## End(Not run)
```
Index

*Topic **BLACS**
  base.mintxt, 11
  base.ownany, 13
  blacsexit, 32
  gridexit, 36
  gridinfo, 37
  InitGrid, 39
  pcoords, 41

*Topic **Data**
  base.ownany, 13

*Topic **Distributing**
  base.ownany, 13

*Topic **Package**
  pbdBASE-package, 3

*Topic **datasets**
  BASE Global Environment, 4
  .pbdBASEEnv (BASE Global Environment), 4
  BASE Global Environment, 4
  base.blacs(gridinfo), 37
  base.blacs_gridinit(gridinit), 38
  base.blacsexit(blacsexit), 32
  base.crossprod, 4
  base.dallreduce, 5
  base.ddiagmk(diag), 34
  base.ddiagtk(diag), 34
  base.descinit, 5
  base.dgamm2d(base.dgamx2d), 6
  base.dgamx2d, 6
  base.dgerv2d(base.dgesd2d), 6
  base.dgesd2d, 6
  base.dgsum2d(base.igsum2d), 9
  base.dhilbmk, 7
  base.dim0, 8
  base.finalize(finalizer), 35
  base.g2l_coord(g2l_coord), 36
  base.gridexit(gridexit), 36
  base.igamm2d(base.dgamx2d), 6
  base.igamx2d, 8
  base.igsum2d, 9
  base.indxg2p, 9
  base.12g_coord(12g_coord), 40
  base.matexp, 10
  base.maxdim, 10
  base.mintxt, 11
  base.mkgb1mat(base.mksubmat), 11
  base.mksubmat, 11
  base.nbd, 12
  base.numroc, 12
  base.ownany, 13
  base.p_matexp_pade_wrap, 17
  base.p_matpow_by_squaring_wrap, 18
  base.pcoord(pcoords), 41
  base.pdhchtri, 14
  base.pdclvar, 14
  base.pdhilbmk, 15
  base.pdmkcpn1, 15
  base.pdmvsum, 16
  base.pdsweep, 16
  base.pnum(pcoords), 41
  base.proccgrid, 17
  base.rcolcpy, 18
  base.rcolcpy2, 19
  base.redist, 19
  base.rldblas, 20
  base.rld2insert, 20
  base.rpdgecon, 21
  base.rpdgels, 21
  base.rpdgemm, 22
  base.rpdgemr2d, 22
  base.rpdgeqpf, 23
  base.rpdgesv, 23
  base.rpdgesvd, 24
  base.rpdgetrf, 24
  base.rpdgetri, 25
  base.rpdllange, 25
  base.rpdlapnt, 26
  base.rpdorgqr, 26
  base.rpdormqr, 27
base.rpdpotrf, 27
base.rpdsvyevr, 28
base.rpdsvyevx, 28
base.rpdtran, 29
base.rpdtrcon, 30
base.rrowcpy, 30
base.rrowcpy2, 31
base.tri2zero, 31
base.valid_context, 32
blacs(gridinfo), 37
blacs_gridinit(gridinit), 38
blacsexit, 32

cords, 33
cordinate, 34
dia, 34

finalize(finalizer), 35
finalizer, 35
g2l_coord, 36
g2lcoord, 35
g2lpair(coordinate), 34
gridexit, 36
gridinfo, 37
gridinit, 38

indexg2l(coords), 33
indexl2g(coords), 33
init.grid(InitGrid), 39
InitGrid, 39

l2g_coord, 40
l2gpair(coordinate), 34

numroc2, 41

pbdBASE-package, 3
pcoords, 41