scalapack_wrapper

NAME

scalapack_wrapper - a frovedis module provides user-friendly interfaces for commonly used scalapack routines in scientific applications like machine learning algorithms.

SYNOPSIS

import frovedis.matrix.wrapper.SCALAPACK

WRAPPER FUNCTIONS

SCALAPACK.getrf (m)
SCALAPACK.getri (m, ipivPtr)
SCALAPACK.getrs (m1, m2, ipivPtr, trans=False)
SCALAPACK.gesv (m1, m2)
SCALAPACK.gels (m1, m2, trans=False)
SCALAPACK.gesvd (m, wantU=False, wantV=False)

DESCRIPTION

ScaLAPACK is a high-performance scientific library written in Fortran language. It provides rich set of linear algebra functionalities whose computation loads are parallelized over the available processes in a system and the user interfaces of this library is very detailed and complex in nature. It requires a strong understanding on each of the input parameters, along with some distribution concepts.

Frovedis provides a wrapper module for some commonly used ScaLAPACK subroutines in scientific applications like machine learning algorithms. These wrapper interfaces are very simple and user needs not to consider all the detailed distribution parameters. Only specifying the target vectors or matrices with some other parameters (depending upon need) are fine. At the same time, all the use cases of a ScaLAPACK routine can also be performed using Frovedis ScaLAPACK wrapper of that routine.

This python module implements a client-server application, where the python client can send the python matrix data to frovedis server side in order to create blockcyclic matrix at frovedis server and then python client can request frovedis server for any of the supported ScaLAPACK operation on that matrix. When required, python client can request frovedis server to send back the resultant matrix and it can then create equivalent python data (see manuals for FrovedisBlockcyclicMatrix to python data conversion).

The individual detailed descriptions can be found in the subsequent sections. Please note that the term "inout", used in the below section indicates a function argument as both "input" and "output".

Detailed Description

getrf (m)

Parameters
m: A FrovedisBlockcyclicMatrix (inout)

Purpose
It computes an LU factorization of a general M-by-N distributed matrix, "m" using partial pivoting with row interchanges.

On successful factorization, matrix "m" is overwritten with the computed L and U factors. Along with the return status of native scalapack routine, it also returns the proxy address of the node local vector "ipiv" containing the pivoting information associated with input matrix "m" in the form of GetrfResult. The "ipiv" information will be useful in computation of some other routines (like getri, getrs etc.)

Return Value
On success, it returns the object of the type GetrfResult as explained above. If any error occurs, it throws an exception explaining cause of the error.

getri (m, ipivPtr)

Parameters
m: A FrovedisBlockcyclicMatrix (inout)
ipiv: A long object containing the proxy of the ipiv vector (from GetrfResult) (input)

Purpose
It computes the inverse of a distributed square matrix using the LU factorization computed by getrf(). So in order to compute inverse of a matrix, first compute it's LU factor (and ipiv information) using getrf() and then pass the factored matrix, "m" along with the "ipiv" information to this function.

On success, factored matrix "m" is overwritten with the inverse (of the matrix which was passed to getrf()) matrix. "ipiv" will be internally used by this function and will remain unchanged.

For example,

```
res = SCALAPACK.getrf(m)      // getting LU factorization of "m"
SCALAPACK.getri(m,res.ipiv()) // "m" will have inverse of the initial value
```

Return Value
On success, it returns the exit status of the scalapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

getrs (m1, m2, ipiv, trans=False)

Parameters
m1: A FrovedisBlockcyclicMatrix (input)
m2: A FrovedisBlockcyclicMatrix (inout)
ipiv: A long object containing the proxy of the ipiv vector (from GetrfResult) (input)
trans: A boolean value to specify whether to transpose "m1" [Default: False] (input/optional)

Purpose
It solves a real system of distributed linear equations, AX=B with a general distributed square matrix (A) using the LU factorization computed by getrf(). Thus before calling this function, it is required to obtain the factored matrix "m1" (along with "ipiv" information) by calling getrf().

For example,

```
res = SCALAPACK.getrf(m1) // getting LU factorization of "m1"
SCALAPACK.getrs(m1,m2,res.ipiv())
```

If trans=False, the linear equation AX=B is solved.
If trans=True, the linear equation transpose(A)X=B (A'X=B) is solved.

The matrix "m2" should have number of rows >= the number of rows in "m1" and at least 1 column in it.

On entry, "m2" contains the distributed right-hand-side (B) of the equation and on successful exit it is overwritten with the distributed solution matrix (X).

Return Value
On success, it returns the exit status of the scalapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gesv (m1, m2)

Parameters
m1: A FrovedisBlockcyclicMatrix (inout)
m2: A FrovedisBlockcyclicMatrix (inout)

Purpose
It solves a real system of distributed linear equations, AX=B with a general distributed square matrix, "m1" by computing it's LU factors internally. This function internally computes the LU factors and ipiv information using getrf() and then solves the equation using getrs().

The matrix "m2" should have number of rows >= the number of rows in "m1" and at least 1 column in it.

On entry, "m1" contains the distributed left-hand-side square matrix (A), "m2" contains the distributed right-hand-side matrix (B) and on successful exit "m1" is overwritten with it's LU factors, "m2" is overwritten with the distributed solution matrix (X).

Return Value
On success, it returns the exit status of the scalapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gels (m1, m2, trans=False)

Parameters
m1: A FrovedisBlockcyclicMatrix (input)
m2: A FrovedisBlockcyclicMatrix (inout)
trans: A boolean value to specify whether to transpose "m1" [Default: False] (input/optional)

Purpose
It solves overdetermined or underdetermined real linear systems involving an M-by-N distributed matrix (A) or its transpose, using a QR or LQ factorization of (A). It is assumed that distributed matrix (A) has full rank.

If trans=False and M >= N: it finds the least squares solution of an overdetermined system.
If trans=False and M < N: it finds the minimum norm solution of an underdetermined system.
If trans=True and M >= N: it finds the minimum norm solution of an underdetermined system.
If trans=True and M < N: it finds the least squares solution of an overdetermined system.

The matrix "m2" should have number of rows >= max(M,N) and at least 1 column.

On entry, "m1" contains the distributed left-hand-side matrix (A) and "m2" contains the distributed right-hand-side matrix (B). On successful exit, "m1" is overwritten with the QR or LQ factors and "m2" is overwritten with the distributed solution matrix (X).

Return Value
On success, it returns the exit status of the scalapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gesvd (m, wantU=False, wantV=False)

Parameters
m: A FrovedisBlockcyclicMatrix (inout)
wantU: A boolean value to specify whether to compute U matrix [Default: False] (input)
wantV: A boolean value to specify whether to compute V matrix [Default: False] (input)

Purpose
It computes the singular value decomposition (SVD) of an M-by-N distributed matrix.

On entry "m" contains the distributed matrix whose singular values are to be computed.

If wantU = wantV = False, then it computes only the singular values in sorted oder, so that sval(i) >= sval(i+1). Otherwise it also computes U and/or V (left and right singular vectors respectively) matrices.

On successful exit, the contents of "m" is destroyed (internally used as workspace).

Return Value
On success, it returns the object of the type GesvdResult containing the singular values and U and V components (based on the requirement) along with the exit status of the native scalapack routine. If any error occurs, it throws an exception explaining cause of the error.

SEE ALSO

blockcyclic_matrix, pblas_wrapper, arpack_wrapper, getrf_result, gesvd_result