2.7 Parallel Basic Linear Algebra Subprograms (PBLAS)

As mentioned above, the character _ should be read as "of the distributed matrix". Let *A* be a generic term for any 2D block cyclically distributed matrix. Its description vector is DESC_A. The values of $LOC_p()$and $LOC_q()$ may be determined via a call to the ScaLAPACK tool function, `NUMROC` as following:

LOCp(M)=NUMROC(M,MB_A,MYROW,RSRC_A,NPROW),

LOCq(N)=NUMROC(N,NB_A,MYCOL,CSRC_A,NPCOL).

Because vectors may be seen as particular matrices, a distributed vector is considered to be a distributed matrix. In this paragraph we will present the syntaxes of the most-used PBLAS routines.

2.7.1 PBLAS routines for vector-vector operations (Level 1)

```
PvDOT(N,DOT,X,IX,JX,DESCX,INCX,Y,IY,JY,DESCY,INCY)
pddot_(int*,double*,double*,int*,int*,int*,int*,double*,int*Y,
int*,int*,int*);
```

Purpose: PvDOT forms the dot product of two distributed vectors, *dot:=sub(X)$^T \times$ sub(Y),* where

$$\mathrm{sub}(X) = \begin{cases} X(IX, JX : JX + N - 1) & \text{if } INCX = M\_X \\ X(IX : IX + N - 1, JX) & \text{if } INCX = 1 \text{ and } INCX <> M\_X \end{cases}$$

$$\mathrm{sub}(Y) = \begin{cases} Y(IY, JY : JY + N - 1) & \text{if } INCY = M\_Y \\ Y(IY : IY + N - 1, JY) & \text{if } INCY = 1 \text{ and } INCY <> M\_Y \end{cases}$$

Arguments

`N (global input) INTEGER`

The length of the distributed vectors to be multiplied. N≥=0.

`DOT (local output) REAL`

The dot product of sub(X) and sub(Y) only in their scope.

`X (local input/local output) array of dimension ((JX-1)*M_X+IX+(N-1)*abs(INCX))`

This array contains the entries of the distributed vector sub(X).

`IX (global input) INTEGER`

The global row index of the submatrix of the distributed matrix X to operate on.

`JX (global input) INTEGER`

The global column index of the submatrix of the distributed matrix X to operate on.

`DESCX (global and local input) INTEGER array of dimension 8`

The array descriptor of the distributed matrix X.

`INCX (global input) INTEGER`

The global increment for the elements of X. Only two values of INCX are supported in this version, namely 1 and M_X.

`Y (local input/local output) array of dimension ((JY-1)*M_Y+IY+(N-1)*abs(INCY))`

This array contains the entries of the distributed vector sub(Y).

`IY (global input) INTEGER`

The global row index of the submatrix of the distributed matrix Y to operate *on.*

`JY (global input) INTEGER`

The global column index of the submatrix of the distributed matrix Y to operate on.

`DESCY (global and local input) INTEGER array of dimension 8`

The array descriptor of the distributed matrix Y.

`INCY (global input) INTEGER`

The global increment for the elements of Y. Only two values of INCY are supported in this version, namely 1 and M_Y.

Parallel Dot Product – Coarse Grain Algorithm

- Combine k=n/p components of the vectors X and Y on each processor to form a coarse-grain task, for a total of p processors,
- Each processor computes the inner product of the corresponding sub-vectors (local inner product),
- Communicate over coarse-grain tasks to compute the sum reduction.

```
PvNRM2(N,NORM2,X,IX,JX,DESCX,INCX)
void pdnrm2_ (int *N, double *NORM2, double *X, int *IX, int *JX, int *DESCX, int *INCX)
```

Purpose: PvNRM2 returns the 2-norm of a distributed vector *sub(X),*

$$\text{sub}(X) = \begin{cases} X(IX, JX : JX + N - 1) & \text{if } INCX = M\_X \\ X(IX : IX + N - 1, JX) & \text{if } INCX = 1 \text{ and } INCX <> M\_X \end{cases}$$

Arguments

`N (global input) INTEGER`

The length of the distributed vectors to be multiplied. N >= 0.

`NORM2 (local output) REAL/COMPLEX`

The dot product of sub(X) and sub(Y) only in their scope..

`X (local input/local output) REAL/COMPLEX array of dimension ((JX-1)*M_X+IX+(N-1)*abs(INCX))`

This array contains the entries of the distributed vector sub(X).

`IX (global input) INTEGER`

The global row index of the submatrix of the distributed matrix X to operate on.

`JX (global input) INTEGER`

The global column index of the submatrix of the distributed matrix X to operate on.

`DESCX (global and local input) INTEGER array of dimension 9`

The array descriptor of the distributed matrix X.

`INCX (global input) INTEGER`

The global increment for the elements of X. Only two values of INCX are supported in this version, namely 1 and M_X (the number of rows in the distributed matrix X, M_X>= 0)

```
PvAMAX(N,AMAX,INDX,X,IX,JX,DESCX,INCX)
void pdamax_(int *N,double *AMAX,int *INDX,double *X,int *IX,
             int *JX,int *DESCX,int *INCX)
```

Purpose: PvAMAX computes the global index of the maximum element in absolute value of a distributed vector *sub(X)*. The global index is returned in `INDX` and the value is returned in AMAX, where

$$\text{sub}(X) = \left\{ \begin{array}{ll} X(IX, JX : JX + N - 1) & \text{if } INCX = M\_X \\ X(IX : IX + N - 1, JX) & \text{if } INCX = 1 \text{ and } INCX <> M\_X \end{array} \right\}$$

Arguments

N (global input) INTEGER

The length of the distributed vectors to be multiplied. N >= 0. .

AMAX (global output) REAL/COMPLEX

The absolute value of the largest entry of the distributed vector sub(X) only in the scope of sub(X).

INDX (global output) INTEGER

The global index of the maximum element in absolute value of the distributed vector sub(X) only in its scope.

X (local input/local output) REAL/COMPLEX array of dimension ((JX-1)*M_X + IX + (N - 1)*abs(INCX))

This array contains the entries of the distributed vector sub(X).

IX (global input) INTEGER

The global row index of the submatrix of the distributed matrix X to operate on.

JX (global input) INTEGER

The global column index of the submatrix of the distributed matrix X to operate on.

DESCX (global and local input) INTEGER array of dimension 8

The array descriptor of the distributed matrix X.

INCX (global input) INTEGER

The global increment for the elements of X. Only two values of INCX are supported in this version, namely 1 and M_X.

```
PvCOPY(N, X, IX, JX, DESCX, INCX, Y, IY, JY, DESCY, INCY )
void pdcopy_(int *N,double *X,int *IX,int *JX,int *DESCX,
              int *INCX,double *Y,int *IY,int *JY,int *DESCY,
              int *INCY)
```

Purpose: PvCOPY copies one distributed vector into another: sub(Y) := sub(X), where

$$\text{sub}(X) = \left\{ \begin{array}{ll} X(IX, JX : JX + N - 1) & \text{if } INCX = M\_X \\ X(IX : IX + N - 1, JX) & \text{if } INCX = 1 \text{ and } INCX <> M\_X \end{array} \right\}$$

$$\text{sub}(Y) = \left\{ \begin{array}{ll} Y(IY, JY : JY + N - 1) & \text{if } INCY = M\_Y \\ X(IY : IY + N - 1, JY) & \text{if } INCY = 1 \text{ and } INCY <> M\_Y \end{array} \right\}$$

Arguments

N (global input) INTEGER

The length of the distributed vectors to be copied. N >= 0.

X (local input/local output) array of dimension ((JX-1)*M_X+IX+(N-1)*abs(INCX))

This array contains the entries of the distributed vector sub(X).

IX (global input) INTEGER

The global row index of the submatrix of the distributed matrix X to operate on.

JX (global input) INTEGER

The global column index of the submatrix of the distributed matrix X to *operate on.*

DESCX (global and local input) INTEGER array of dimension 8

The array descriptor of the distributed matrix X.

INCX (global input) INTEGER

The global increment for the elements of X. Only two values of INCX are supported in this version, namely 1 and M_X.

Y (local input/local output) array of dimension ((JY-1)*M_Y + IY + (N - 1)*abs(INCY))

This array contains the entries of the distributed vector sub(Y). On exit sub(Y) is overwritten by sub(X).

IY(global input) INTEGER

The global row index of the submatrix of the distributed matrix Y to operate on.

JY (global input) INTEGER

The global column index of the submatrix of the distributed matrix Y to operate on.

DESCY (global and local input) INTEGER array of dimension 8

The array descriptor of the distributed matrix Y.

INCY (global input) INTEGER

The global increment for the elements of Y. Only two values of INCY are supported in this version, namely 1 and M_Y.

Example 2.7.1. *An example of using the functions PBLAS Level 1.*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "mpi.h"
extern "C" {
/* Cblacs declarations */
void Cblacs_pinfo(int*, int*);
void Cblacs_get(int, int, int*);
void Cblacs_gridinit(int*, const char*, int, int);
void Cblacs_gridinfo(int,int*,int*,int*,int*);
void Cblacs_gridexit(int);
void descinit_(int*,int*,int*,int*,int*,int*,int*,int*,int*,int*);
void
pddot_(int*,double*,double*,int*,int*,int*,int*,double*,int*Y,int*,int*,int*);
void pdnrm2_(int*,double*,double*,int*,int*,int*,int*);
void pdamax_(int*,double*,int*,double*,int*,int*,int*,int*);
void
pdcopy_(int*,double*,int*,int*,int*,int*,double*,int*,int*,int*,int*);
int numroc_(int*, int*, int*, int*, int*);
void Cblacs_barrier(int,const char*);
}
/*******************************************************
*
Globals
*
*******************************************************/
int N=10, ZERO=0, ONE=1;; //N-dimensiunea globala(initiala) a vectorilor
main(int argc, char* argv[]) {
int myid;
/* process rank */
int p;
/* number of processes */
int n, nb, i, i0, INDX;
int ctxt, prow, pcol, myrow, mycol;
/* BLACS stuff */
int descx[9], descy[9], info;
int size = N * sizeof(double);
double s, *x, *y,norm2,AMAX;
/* Start the MPI engine */
MPI_Init(&argc, &argv);
/* Find out number of processes */
MPI_Comm_size(MPI_COMM_WORLD, &p);
size = (size + p - 1)/p; // dimensiunea locala a vectorilor
x = (double*) malloc(size);
y = (double*) malloc(size);
/* Find out process rank */
MPI_Comm_rank(MPI_COMM_WORLD, &myid);
if (myid ==0)
printf("=====Rezultatul programului,%s \n",argv[0]);
/* Get a BLACS context */
Cblacs_get(0, 0, &ctxt);
/* Initialize a 1-dimensional grid */
Cblacs_gridinit(&ctxt, "R", p, 1);
Cblacs_gridinfo(ctxt, &prow, &pcol, &myrow, &mycol);
nb = (N + p - 1)/p;
/* nb = ceil(N/p) */ //dimensiunea blocului
i0 = myid*nb;
n = nb;
if (myid == p-1) n = N - i0;
//printf("Lungimea vectorului (valoarea lui n) pentru procesul %d este %d
\n",myid,n);
int nrows = numroc_(&N, &nb, &myrow, &ZERO, &prow);
printf("Lungimea vectorului (valoarea lui nrows) pentru procesul %d este %d
\n",myid,nrows);
descinit_(descx,&N,&ONE,&nb,&ONE,&ZERO,&ZERO,&ctxt,&nrows,&info);
descinit_(descy,&N,&ONE,&nb,&ONE,&ZERO,&ZERO,&ctxt,&nrows,&info);
/* Initialize vectors */
for (i = 0; i < n; i++){
x[i] = myid;//rand()/100.0;//(double)i;
y[i] = 1.0;//(double)i0 + (double)i;
}
/* Check result */
s = 0.0;
// se determina produsul scalar a vectorilor "globali"
pddot_(&N,&s,x,&ONE,&ONE,descx,&ONE,y,&ONE,&ONE,descy,&ONE);
// se determina norma vectorului "global"
pdnrm2_ (&N, &norm2, x, &ONE, &ONE, descx, &ONE);
// se determina elementul maximal si indicele lui pentru vectorul "global"
pdamax_(&N,&AMAX,&INDX,x,&ONE, &ONE,descx,&ONE);
//se copie subvectorul X in subvectorul Y
pdcopy_(&N,x,&ONE,&ONE,descx,&ONE,y,&ONE,&ONE,descy,&ONE);
if (myid == 0)
{
printf("Dot produs of the vector X and Y is = %10.2f \n", s);
printf("Norm of the vector X is = %10.2f \n", norm2);
printf("The absolute maximum of X is = %10.2f and global index of the maximum is =
%d \n", AMAX,INDX);
}

for (i = 0; i < n; i++){
printf("For process (%d,%d) the vector Y[%d] is = %5.2f \n", myrow,mycol,i,y[i]);
                  }
```

```
/* Release process grid */
Cblacs_gridexit(ctxt);
/* Shut down MPI */
MPI_Finalize();
} /* main */
```

The results are the following

```
[MI_gr_TPS1@hpc]$./mpiCC_ScL -o Example2.7.1.exe Example2.7.1.cpp
[MI_gr_TPS1@hpc]$ opt/openmpi/bin/mpirun -n 3 -host compute-0-0,compute-0-1
Example2.7.1.exe
Lungimea vectorului (valoarea lui nrows) pentru procesul 0 este 34
Lungimea vectorului (valoarea lui nrows) pentru procesul 1 este 34
Lungimea vectorului (valoarea lui nrows) pentru procesul 2 este 32
For process (2,0) the vector Y[0] is =  2.00
Dot produs of the vector X and Y is =      98.00
Norm of the vector X is =      12.73
The absolute maximum of X is =       2.00 and global index of the maximum is = 69
For process (1,0) the vector Y[0] is =  1.00
For process (1,0) the vector Y[1] is =  1.00
For process (2,0) the vector Y[1] is =  2.00
For process (2,0) the vector Y[2] is =  2.00
For process (1,0) the vector Y[2] is =  1.00
For process (1,0) the vector Y[3] is =  1.00
For process (1,0) the vector Y[4] is =  1.00
For process (1,0) the vector Y[5] is =  1.00
For process (1,0) the vector Y[6] is =  1.00
For process (0,0) the vector Y[0] is =  0.00
For process (0,0) the vector Y[1] is =  0.00
For process (0,0) the vector Y[2] is =  0.00
For process (0,0) the vector Y[3] is =  0.00
For process (0,0) the vector Y[4] is =  0.00
For process (1,0) the vector Y[7] is =  1.00
For process (1,0) the vector Y[8] is =  1.00
For process (1,0) the vector Y[9] is =  1.00
For process (1,0) the vector Y[10] is =  1.00
For process (2,0) the vector Y[3] is =  2.00
For process (2,0) the vector Y[4] is =  2.00
For process (2,0) the vector Y[5] is =  2.00
For process (2,0) the vector Y[6] is =  2.00
For process (2,0) the vector Y[7] is =  2.00
For process (1,0) the vector Y[11] is =  1.00
For process (1,0) the vector Y[12] is =  1.00
For process (1,0) the vector Y[13] is =  1.00
For process (0,0) the vector Y[5] is =  0.00
For process (0,0) the vector Y[6] is =  0.00
For process (0,0) the vector Y[7] is =  0.00
For process (0,0) the vector Y[8] is =  0.00
For process (1,0) the vector Y[14] is =  1.00
For process (1,0) the vector Y[15] is =  1.00
For process (1,0) the vector Y[16] is =  1.00
For process (2,0) the vector Y[8] is =  2.00
For process (2,0) the vector Y[9] is =  2.00
For process (2,0) the vector Y[10] is =  2.00
For process (1,0) the vector Y[17] is =  1.00
For process (1,0) the vector Y[18] is =  1.00
For process (1,0) the vector Y[19] is =  1.00
For process (0,0) the vector Y[9] is =  0.00
For process (0,0) the vector Y[10] is =  0.00
For process (0,0) the vector Y[11] is =  0.00
For process (0,0) the vector Y[12] is =  0.00
For process (0,0) the vector Y[13] is =  0.00
For process (1,0) the vector Y[20] is =  1.00
For process (1,0) the vector Y[21] is =  1.00
For process (1,0) the vector Y[22] is =  1.00
For process (2,0) the vector Y[11] is =  2.00
For process (2,0) the vector Y[12] is =  2.00
For process (2,0) the vector Y[13] is =  2.00
For process (2,0) the vector Y[14] is =  2.00
For process (2,0) the vector Y[15] is =  2.00
For process (2,0) the vector Y[16] is =  2.00
For process (2,0) the vector Y[17] is =  2.00
For process (2,0) the vector Y[18] is =  2.00
For process (2,0) the vector Y[19] is =  2.00
For process (1,0) the vector Y[23] is =  1.00
For process (1,0) the vector Y[24] is =  1.00
For process (1,0) the vector Y[25] is =  1.00
For process (0,0) the vector Y[14] is =  0.00
For process (0,0) the vector Y[15] is =  0.00
For process (0,0) the vector Y[16] is =  0.00
For process (0,0) the vector Y[17] is =  0.00
For process (0,0) the vector Y[18] is =  0.00
For process (0,0) the vector Y[19] is =  0.00
For process (0,0) the vector Y[20] is =  0.00
For process (0,0) the vector Y[21] is =  0.00
For process (0,0) the vector Y[22] is =  0.00
For process (0,0) the vector Y[23] is =  0.00
For process (1,0) the vector Y[26] is =  1.00
For process (1,0) the vector Y[27] is =  1.00
For process (1,0) the vector Y[28] is =  1.00
For process (2,0) the vector Y[20] is =  2.00
For process (2,0) the vector Y[21] is =  2.00
For process (2,0) the vector Y[22] is =  2.00
For process (2,0) the vector Y[23] is =  2.00
For process (2,0) the vector Y[24] is =  2.00
For process (1,0) the vector Y[29] is =  1.00
For process (1,0) the vector Y[30] is =  1.00
For process (1,0) the vector Y[31] is =  1.00
For process (1,0) the vector Y[32] is =  1.00
For process (0,0) the vector Y[24] is =  0.00
For process (0,0) the vector Y[25] is =  0.00
For process (0,0) the vector Y[26] is =  0.00
For process (0,0) the vector Y[27] is =  0.00
For process (1,0) the vector Y[33] is =  1.00
For process (2,0) the vector Y[25] is =  2.00
For process (2,0) the vector Y[26] is =  2.00
For process (2,0) the vector Y[27] is =  2.00
For process (2,0) the vector Y[28] is =  2.00
For process (0,0) the vector Y[28] is =  0.00
For process (0,0) the vector Y[29] is =  0.00
For process (2,0) the vector Y[29] is =  2.00
For process (0,0) the vector Y[30] is =  0.00
For process (0,0) the vector Y[31] is =  0.00
For process (2,0) the vector Y[30] is =  2.00
For process (2,0) the vector Y[31] is =  2.00
For process (0,0) the vector Y[32] is =  0.00
For process (0,0) the vector Y[33] is =  0.00
```

```
[MI_gr_TPS1@hpc]$ /opt/openmpi/bin/mpirun -n 6 -host compute-0-0,compute-0-1
Example2.7.1.exe
Lungimea vectorului (valoarea lui nrows) pentru procesul 5 este 15
Lungimea vectorului (valoarea lui nrows) pentru procesul 0 este 17
Lungimea vectorului (valoarea lui nrows) pentru procesul 1 este 17
Lungimea vectorului (valoarea lui nrows) pentru procesul 2 este 17
Lungimea vectorului (valoarea lui nrows) pentru procesul 3 este 17
Lungimea vectorului (valoarea lui nrows) pentru procesul 4 este 17
Dot produs of the vector X and Y is =     245.00
Norm of the vector X is =      29.75
The absolute maximum of X is =       5.00 and global index of the maximum is = 86

[MI_gr_TPS1@hpc]$ /opt/openmpi/bin/mpirun -n 8 -host compute-0-0,compute-0-1
Example2.7.1.exe
Lungimea vectorului (valoarea lui nrows) pentru procesul 0 este 13
Lungimea vectorului (valoarea lui nrows) pentru procesul 5 este 13
Lungimea vectorului (valoarea lui nrows) pentru procesul 2 este 13
Lungimea vectorului (valoarea lui nrows) pentru procesul 1 este 13
Lungimea vectorului (valoarea lui nrows) pentru procesul 4 este 13
Lungimea vectorului (valoarea lui nrows) pentru procesul 3 este 13
Lungimea vectorului (valoarea lui nrows) pentru procesul 6 este 13
Lungimea vectorului (valoarea lui nrows) pentru procesul 7 este 9
Dot produs of the vector X and Y is =     336.00
Norm of the vector X is =      40.30
The absolute maximum of X is =       7.00 and global index of the maximum is = 92
```

PBLAS Level 1 Routine Groups and Their Data Types

Routine or Function Group	**Data Types**	**Description**
p?amax	s, d, c, z	Calculates an index of the distributed vector element with maximum absolute value
p?asum	s, d, sc, dz	Calculates sum of magnitudes of a distributed vector
p?axpy	s, d, c, z	Calculates distributed vector-scalar product
p?copy	s, d, c, z	Copies a distributed vector
p?dot	s, d	Calculates a dot product of two distributed real vectors
p?dotc	c, z	Calculates a dot product of two distributed complex vectors, one of them is conjugated
p?dotu	c, z	Calculates a dot product of two distributed complex vectors
p?nrm2	s, d, sc, dz	Calculates the 2-norm (Euclidean norm) of a distributed vector
p?scal	s, d, c, z, cs, zd	Calculates a product of a distributed vector by a scalar
p?swap	s, d, c, z	Swaps two distributed vectors