Parallel Processing

Dr. Guy Tel-Zur

Edited. ~~9/12/2013, 7/12/2014,~~ ~~4/5/2015~~, ~~7/12/2015~~, ~~01/05/2016~~, ~~12/12/2016~~, ~~8/4/2019~~ ~~30/11/2019~~, ~~9/5/2020~~, ~~29/11/2020~~, 12/4/2021

Agenda

- More on Jacobi Iterations and the Heat Equation
- MPI Virtual Topologies
- Scalapack
- Mixing programming languages
- Home Assignment #2

Iterative Methods

The following 14 slides are from: “Numerical Methods in Chemical Engineering” by the Laser Analytics Group, University of Cambridge

Reference: http://laser.cheng.cam.ac.uk/wiki/images/d/d1/NumMeth_Handout_4.pdf

- We will discuss more on iterative methods later on in the course

4.1.1 Steady heat conduction in a slab (Laplace's Equation)

The equation which governs temperature in a slab of material is

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T \,, \qquad (4\text{-}1)$$

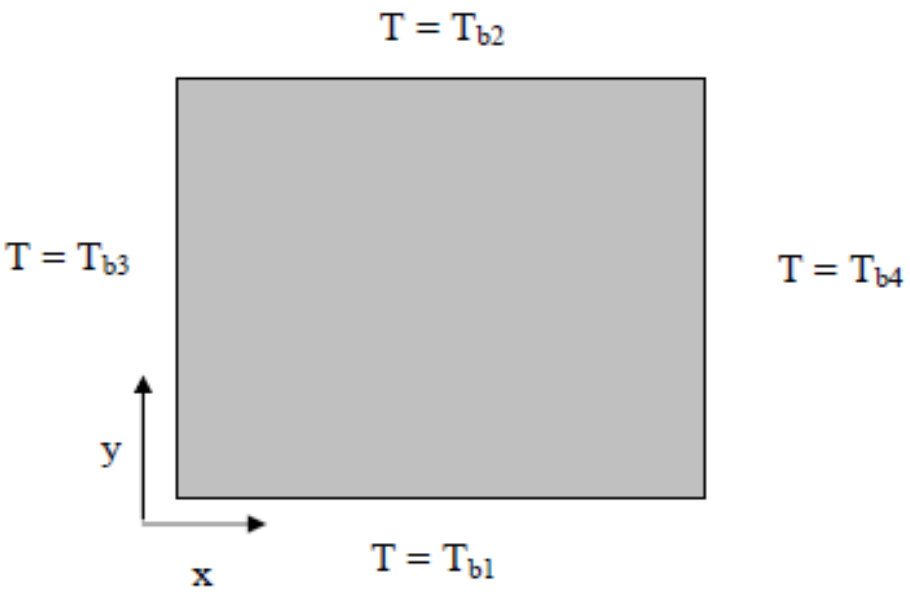
where α is the thermal diffusivity. Here we will only consider the steady state problem, i.e.

$$\nabla^2 T = 0 \,, \qquad (4\text{-}2)$$

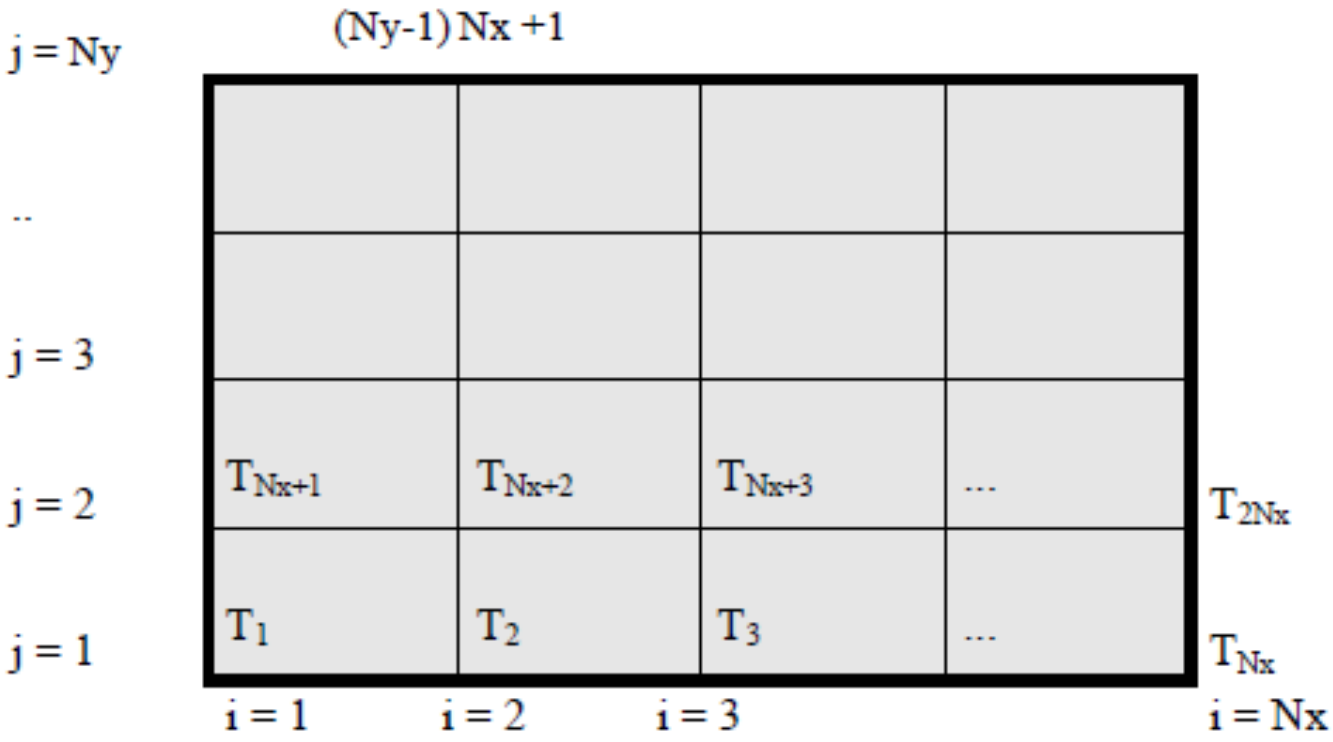
which, in two dimensions and Cartesian co-ordinates can be written as:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \qquad (4\text{-}3)$$

We are going to solve this problem on the following domain,



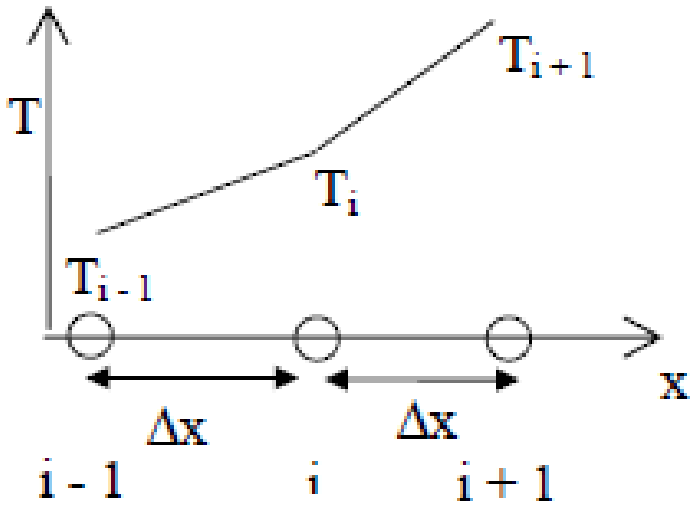
To proceed, we place a grid over our solution domain, and track the temperature at each point on the grid. For simplicity, we will use a regular, equally spaced grid ($\Delta x = \Delta y$).



The index of the general node is

$$k = i + Nx\,(j - 1) \qquad (4\text{-}4)$$

So that $T_{i,j} = T_{k=i+Nx\,(j-1)}$



Then we can estimate the second derivative as

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{\left.\frac{\partial T}{\partial x}\right|_{i+1/2} - \left.\frac{\partial T}{\partial x}\right|_{i-1/2}}{\Delta x}$$

$$\approx \frac{\frac{\left(T_{i+1,j} - T_{i,j}\right)}{\Delta x} - \frac{\left(T_{i,j} - T_{i-1,j}\right)}{\Delta x}}{\Delta x}$$

$$= \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} \qquad (4\text{-}5)$$

Making a similar approximation for the second differential in the y direction allows us to write an approximate form of (4-3) as:

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = 0 \qquad (4\text{-}6)$$

(or in terms of the node indices)

$$\frac{T_{k+1} - 2T_k + T_{k-1}}{\Delta x^2} + \frac{T_{k+Nx} - 2T_k + T_{k-Nx}}{\Delta y^2} = 0 \qquad (4\text{-}7)$$

For this example lets use an equal spaced grid ($\Delta x = \Delta y$)

$$T_{k-Nx} + T_{k-1} - 4T_k + T_{k+1} + T_{k+Nx} = 0 \qquad (4\text{-}8)$$

$$(or\ T_k = (T_{k-Nx} + T_{k-1} + T_{k+1} + T_{k+Nx})/4)$$

The temperature at a grid point is equal to the average of the surrounding temperatures.

For the nodes on the boundary, we have a very simple equation:

$$T_{k,boundary} = \text{some fixed temperature}, \tag{4-9}$$

because, we want to specify the boundary temperatures.

Equations (4-8) for each node and (4-9) for the boundary nodes are linear, and can be written as a matrix equation

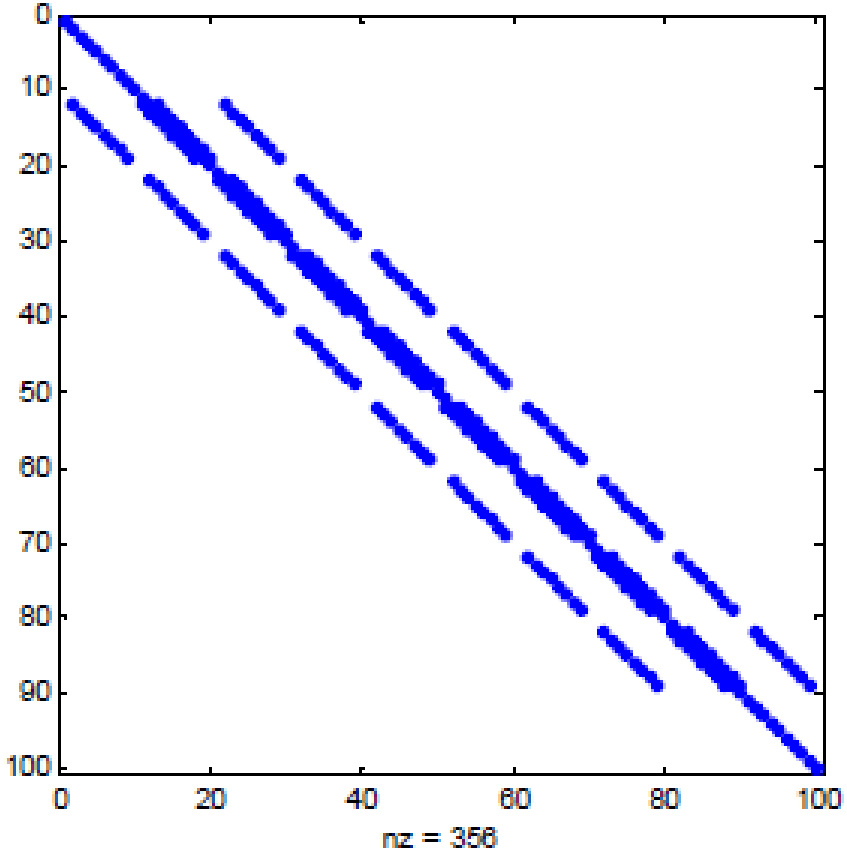
$$A\underline{T} = b \tag{4-10}$$

The matrix A and right hand side vector $\underline{b}$ for $Nx = Ny = 5$ is written out below

A

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Tb1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Tb1
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Tb1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Tb1
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Tb1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Tb3
0	1	0	0	0	1	-4	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	-4	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	-4	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Tb4
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Tb3
0	0	0	0	0	0	1	0	0	0	1	-4	1	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	1	-4	1	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	1	-4	1	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	Tb4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	Tb3
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	-4	1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	-4	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	-4	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	Tb4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	Tb2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	Tb2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	Tb2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	Tb2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Tb2

A plot produced by spy(A) for Nx = Ny = 10



We can see that this sparse matrix has structure, but is not tri-diagonal. There are offset bands. It is these off-set bands which can cause trouble.

4.1.2 Solving the equations using an elimination method

The system equation $A\,\underline{T} = \underline{b}$ we set up in the previous section can be solved in Matlab using the \ command. The solution for $Nx = Ny = 10$ is shown below for $T_{b1} = 10$ and $T_{b2} = T_{b3} = T_{b4} = 0$.

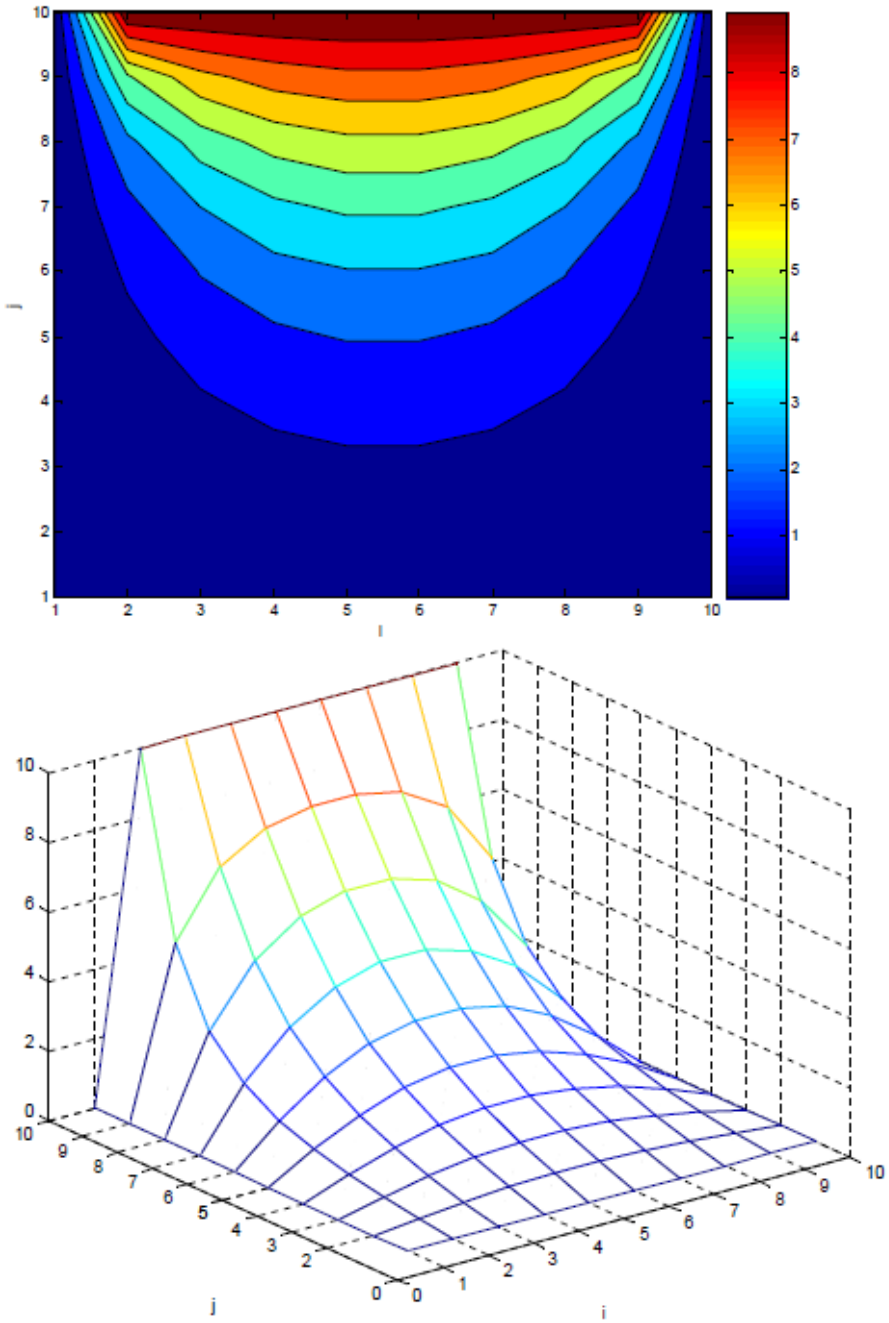
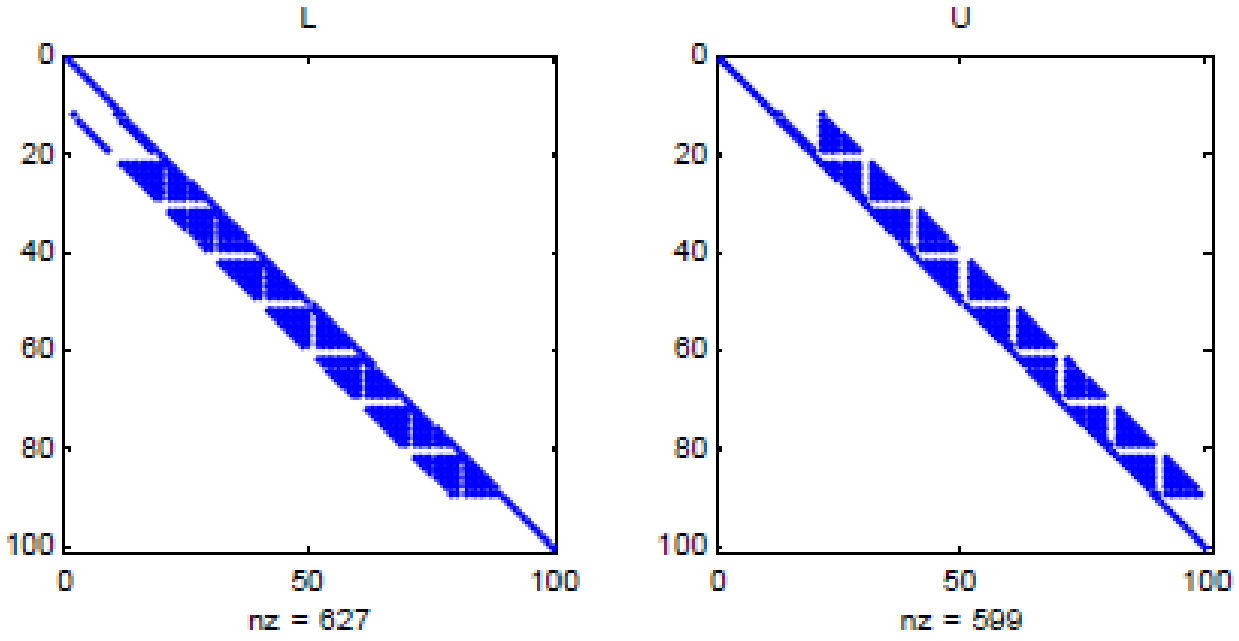


Figure 1: Solution to the Laplace Equation over a square domain. Nx = Ny = 10. All boundaries are at T = 0 except for nodes with j = 10 which is at T = 10.

We can see what happens when we do Gaussian elimination on A by doing LU decomposition (Using the Matlab routine `lu`).

```
[L,U ,P] = lu(A)
subplot(1,2,1)
spy(L)
subplot(1,2,2)
spy(U)
```



When we LU factorise the matrix A we produce matrices, which are less sparse than the original matrices. Now we have filled in all the zero elements between the central band and the offset diagonal. If we do Gaussian elimination on a matrix like A, we will need to store more elements as the algorithm proceeds. If we had solved a 3D problem, we would have had another offset diagonal band even further from the central diagonal band.

Thus, the matrix produced by elimination takes up more memory than the original matrix. For Matlab this is not a problem – it just allocates extra memory. Other languages (where you control memory management) would be less forgiving. The other clever thing Matlab does is attempt to reorder the equations in order to move elements closer to the diagonal.

This problem is sometimes called "fill in". "Fill in" means that when we solve partial differential equations, we will often want to use a method other than e.g. Gaussian elimination. Iterative methods provide an alternative way of solving a set of linear equations.

Typically, we will only use iterative methods when we have large, sparse systems of equations, where the structure of the sparse matrix is not conducive to the use of special methods (e.g. tridiagonal). We will mainly encounter these sorts of systems of equations when we try to solve PDE's in anything other than 1D geometry.

4.2.1 The Jacobi method

In our previous example, the equation derived from node k was

$$T_{k-Nx} + T_{k-1} - 4T_k + T_{k+1} + T_{k+Nx} = 0 \qquad (4\text{-}11)$$

which can be re-arranged as

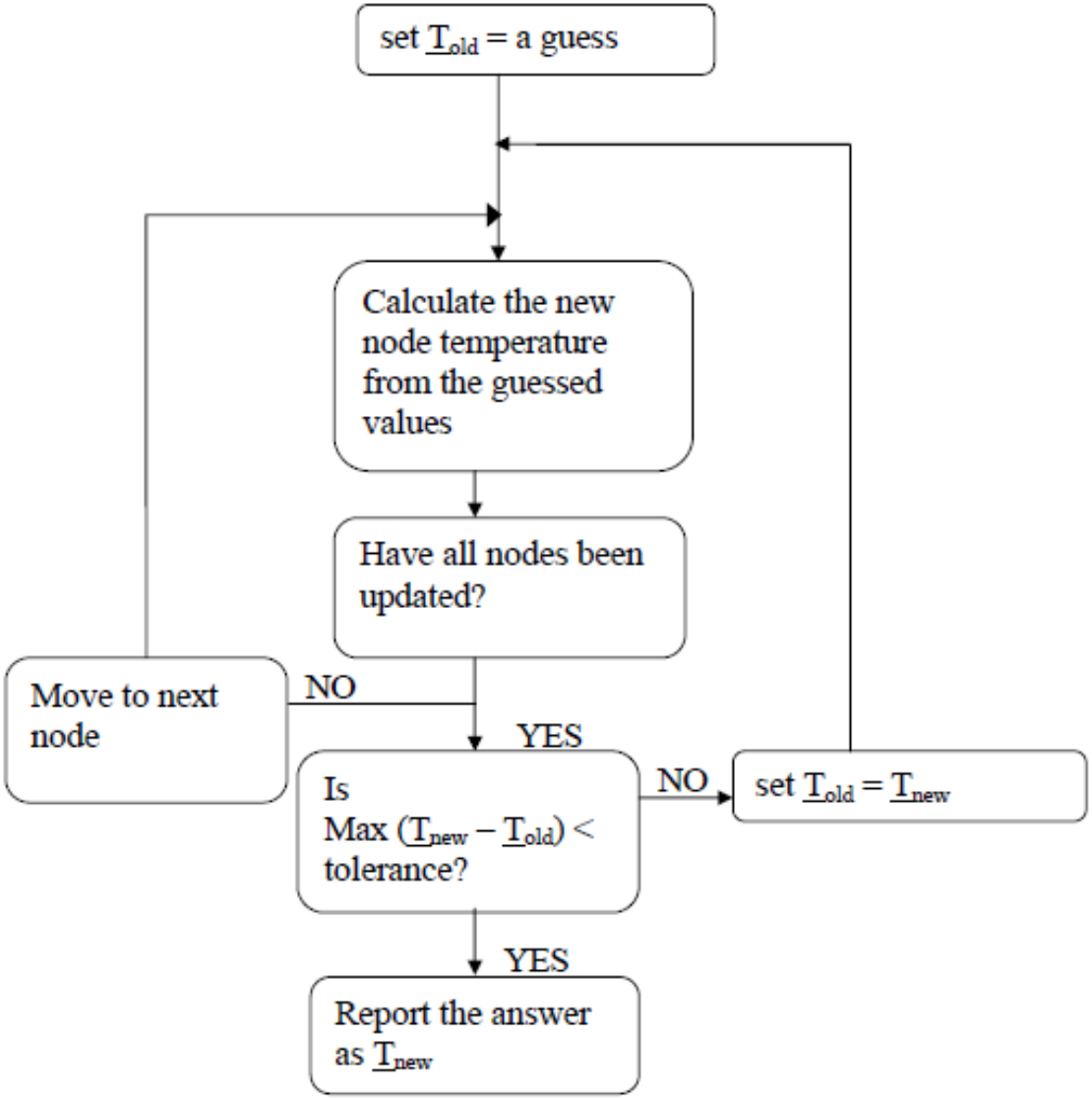
$$T_k = \frac{T_{k-Nx} + T_{k-1} + T_{k+1} + T_{k+Nx}}{4} \qquad (4\text{-}12)$$

In the Jacobi scheme the iteration proceeds as follows:

Starting with an initial guess for the values of T at each node, we form a new updated value using (4-12) and store it in a new vector $\underline{T}_{new}$

$$T_{k,new} = \frac{T_{k-Nx,old} + T_{k-1,old} + T_{k+1,old} + T_{k+Nx,old}}{4} \qquad (4\text{-}13)$$

We then repeat this for all other nodes. Once we have cycled through all nodes, we use the new value as a guess, and repeat.



The advantage here is that only two vectors need to be stored – the old temperature and the updated temperature. This operation can also be performed using matrices:

Laplace Equation in Matlab

Resource: Matlab Central

- http://www.mathworks.com/matlabcentral/fileexchange/38091
- Program 1: Laplace_equation_2D.m
- Program 2: Laplaceimplicit.m

Program 1:
Jacobi iterations

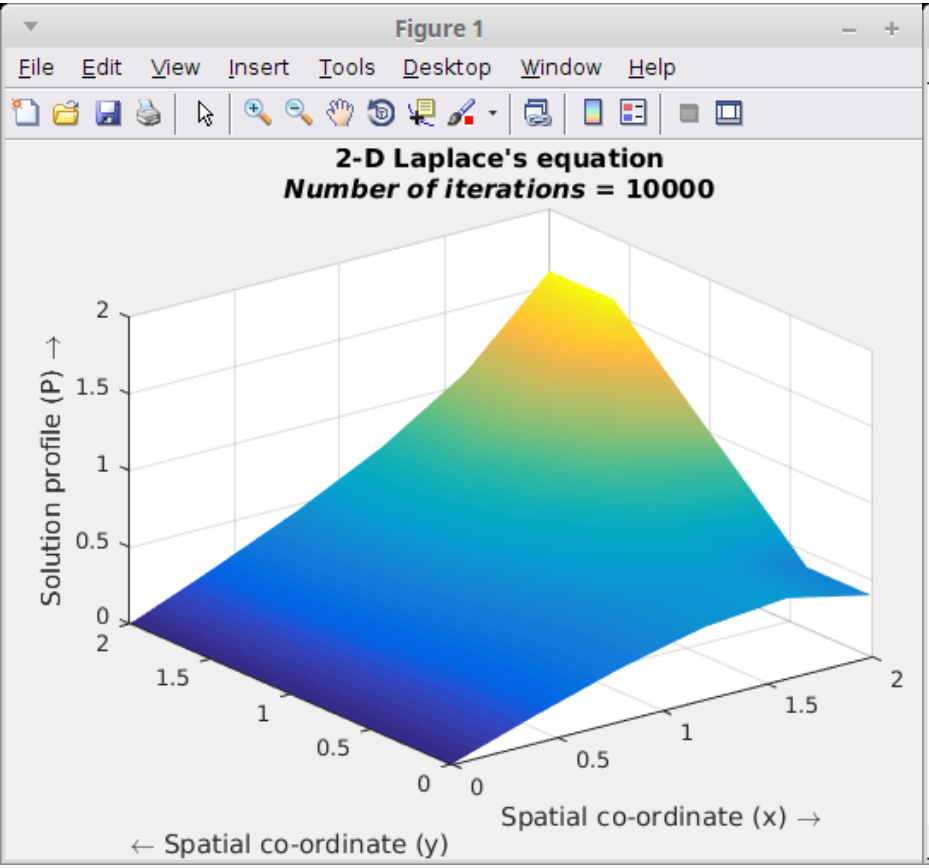
Program: Laplace_equation_2D.m

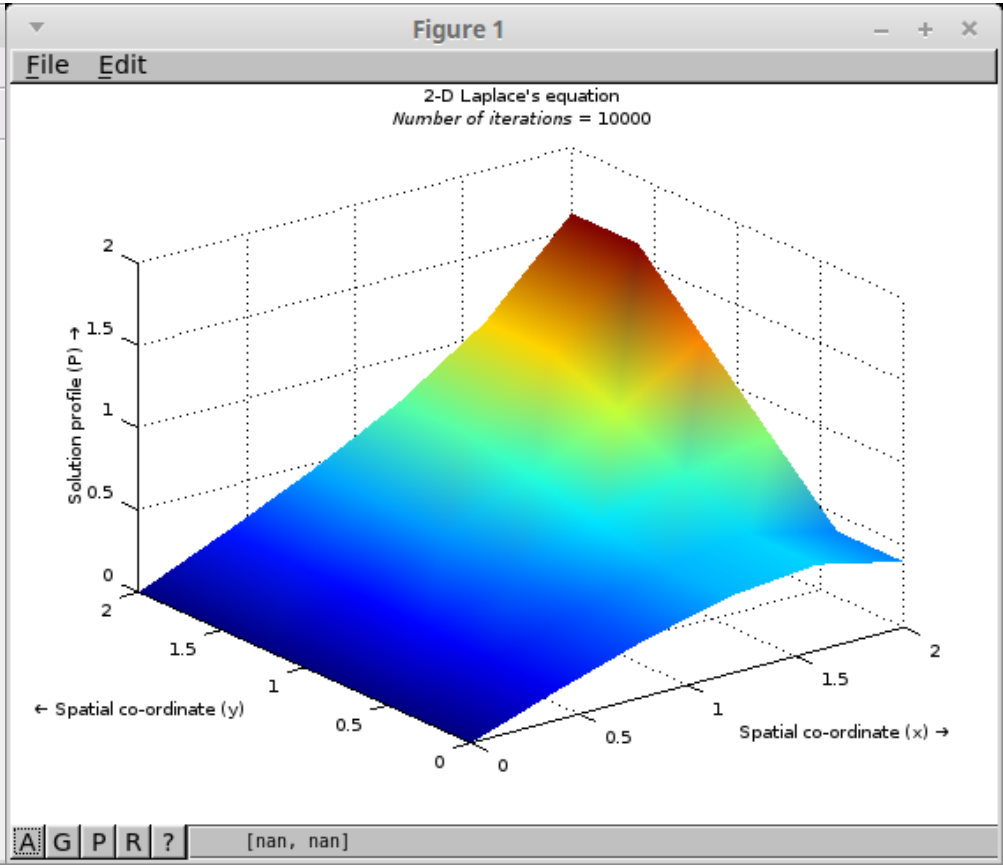
```
% Solving the 2-D Laplace's equation by the Finite Difference
...Method
% Numerical scheme used is a second order central difference
in space
...(5-point difference)

%%
%Specifying parameters
nx=60;                                  %Number of steps in space(x)
ny=60;                                  %Number of steps in space(y)

niter=10000;                            %Number of iterations
dx=2/(nx-1);                            %Width of space step(x)
dy=2/(ny-1);                            %Width of space step(y)
x=0:dx:2;                               %Range of x(0,2) and
specifying the grid points
y=0:dy:2;                               %Range of y(0,2) and
specifying the grid points

%%
%Initial Conditions
p=zeros(ny,nx);                         %Preallocating p
pn=zeros(ny,nx);                        %Preallocating pn
```

```
%%
%Boundary conditions
p(:,1)=0;
p(:,nx)=y;
p(1,:)=p(2,:);                          %Neumann conditions
p(ny,:)=p(ny-1,:);                      ...same as above

%%
%Explicit iterative scheme with C.D in space (5-point
difference)
j=2:nx-1;
i=2:ny-1;
for it=1:niter
    pn=p;
    p(i,j)=((dy^2*(pn(i+1,j)+pn(i-1,j)))+
(dx^2*(pn(i,j+1)+pn(i,j-1))))/(2*(dx^2+dy^2));
    %Boundary conditions (Neumann conditions)
    p(:,1)=0;
    p(:,nx)=y;
    p(1,:)=p(2,:);
    p(ny,:)=p(ny-1,:);
end
```

```
%%
%Plotting the solution
surf(x,y,p,'EdgeColor','none');
shading interp
title({'2-D Laplace''s equation';['{\itNumber of iterations}
= ',num2str(it)]})
xlabel('Spatial co-ordinate (x) \rightarrow')
ylabel('{\leftarrow} Spatial co-ordinate (y)')
zlabel('Solution profile (P) \rightarrow')
```

Plots. Matlab (left) and QtOctave (right)





Program 2:
using matlab '\'

Program: laplaceimplicit.m

```
% Solving the 2-D Laplace's equation by the Finite Difference
...Method
% Numerical scheme used is an implicit second order central
difference in space(5-point difference)

%Specifying parameters
nx=100;                                 %Number of steps in space(x)
ny=100;                                 %Number of steps in space(y)

dx=2/(nx-1);                            %Width of space step(x)
dy=2/(ny-1);                            %Width of space step(y)
x=0:dx:2;                               %Range of x(0,2) and
specifying the grid points
y=0:dy:2;                               %Range of y(0,2) and
specifying the grid points
UW=0;                                   %x=0 Dirichlet B.C
UE=y;                                   %x=L Dirichlet B.C
US=0;                                   %y=0 Dirichlet B.C
UN=0;                                   %y=L Dirichlet B.C
UnW=0;                                  %x=0 Neumann B.C (du/dn=UnW)
UnE=0;                                  %x=L Neumann B.C (du/dn=UnE)
UnS=0;                                  %y=0 Neumann B.C (du/dn=UnS)
UnN=0;                                  %y=L Neumann B.C (du/dn=UnN)
u=zeros(nx,ny);                         %Pre-allocating u
```

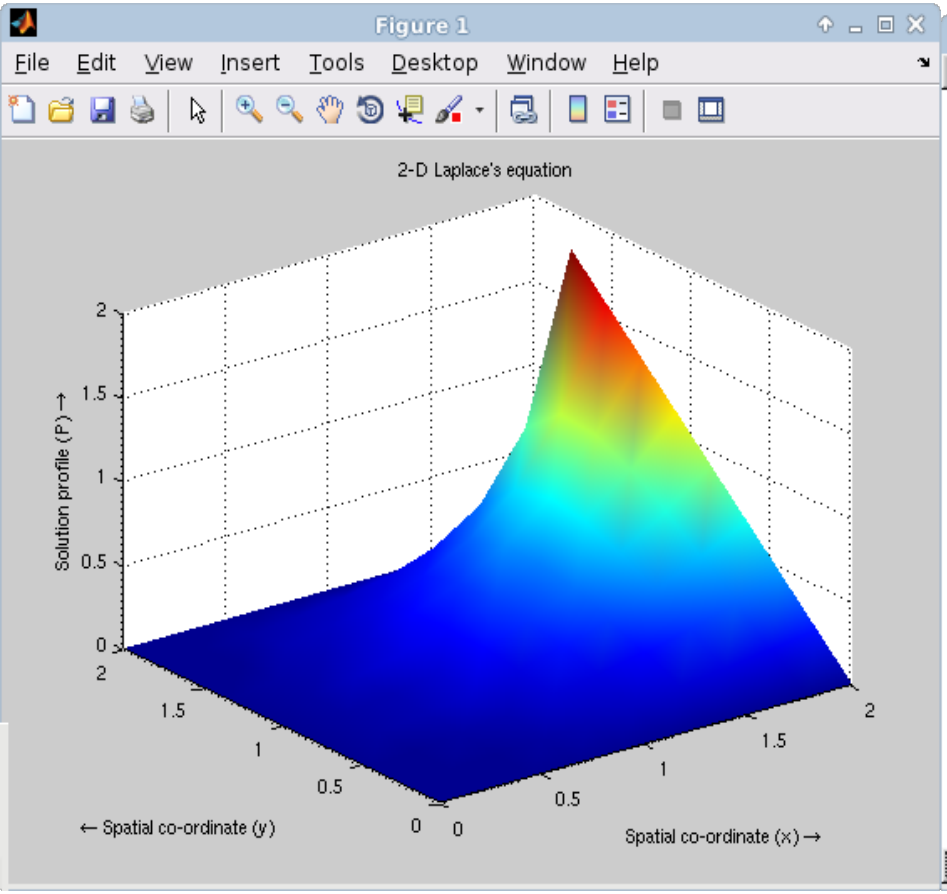
```
%%
%B.C vector
bc=zeros(nx-2,ny-2);
bc(1,:)=UW/dx^2; bc(nx-2,:)=UE(2:ny-1)/dx^2;  %Dirichlet B.Cs
bc(:,1)=-UnS/dy; bc(:,ny-2)=UnN/dy;  %Neumann B.Cs
%B.Cs at the corners:
bc(1,1)=UW/dx^2-UnS/dy; bc(nx-2,1)=UE(2)/dx^2-UnS/dy;
bc(1,ny-2)=UW/dx^2+UnN/dy;
bc(nx-2,ny-2)=UE(ny-1)/dx^2+UnN/dy;

%Calculating the coefficient matrix for the implicit scheme
Ex=sparse(2:nx-2,1:nx-3,1,nx-2,nx-2);
Ax=Ex+Ex'-2*speye(nx-2);          %Dirichlet B.Cs
Ey=sparse(2:ny-2,1:ny-3,1,ny-2,ny-2);
Ay=Ey+Ey'-2*speye(ny-2);          %Dirichlet B.Cs
Ay(1,1)=-1; Ay(ny-2,ny-2)=-1;  %Neumann B.Cs
A=kron(Ay/dy^2,speye(nx-2))+kron(speye(ny-2),Ax/dx^2);
```

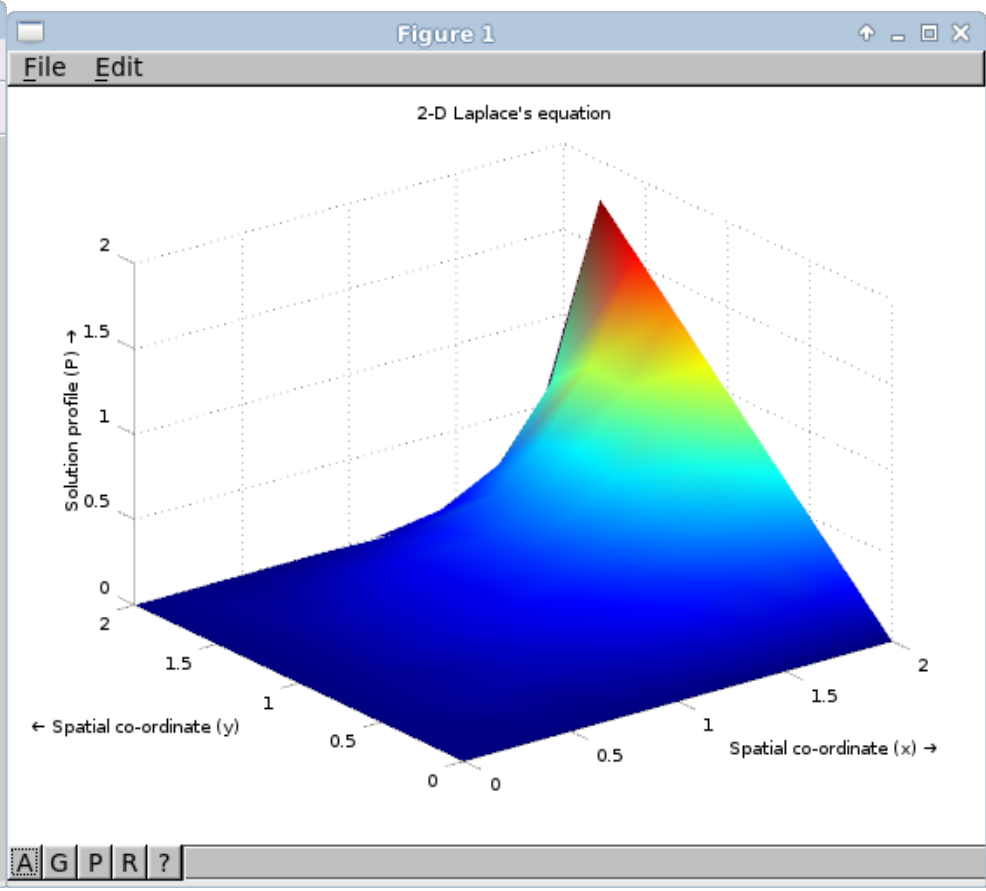
speye=sparse eye (compressed)

```
%%
S=zeros(nx-2,ny-2);                    %Source term
S=reshape(S-bc,[],1);
S=A\S;
S=reshape(S,nx-2,ny-2);
u(2:nx-1,2:ny-1)=S;
%Boundary conditions
%Dirichlet:
u(1,:)=UW;
u(nx,:)=UE;

u(:,1)=u(:,2)-UnS*dy;
u(:,ny)=u(:,ny-1)+UnN*dy;
```

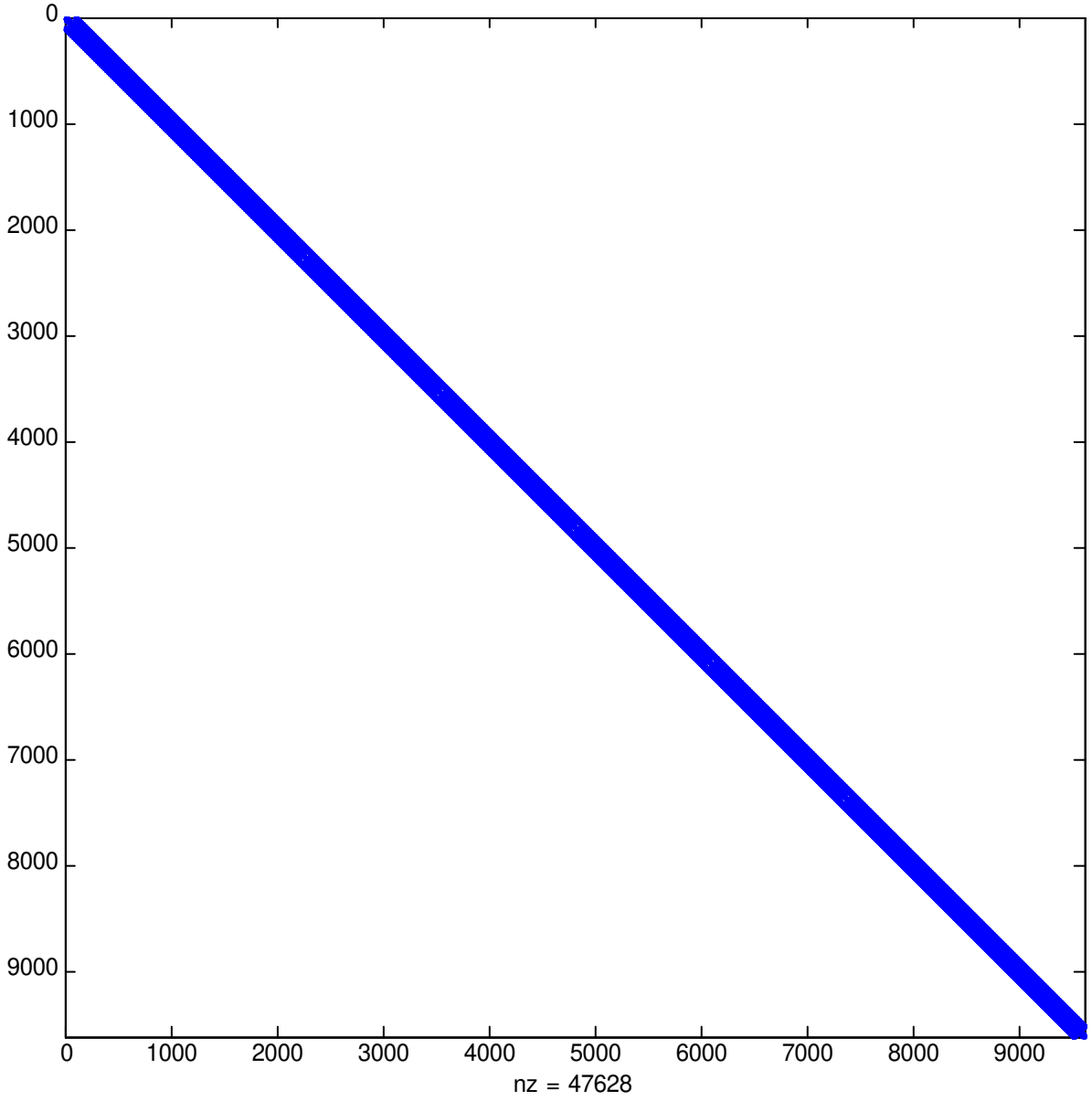
```
%%
%Plotting the solution
surf(x,y,u','EdgeColor','none');
shading interp
title('2-D Laplace''s equation')
xlabel('Spatial co-ordinate (x) \rightarrow')
ylabel('{\leftarrow} Spatial co-ordinate (y)')
zlabel('Solution profile (P) \rightarrow')
```

Plots. Matlab (left), QtOctave (right)

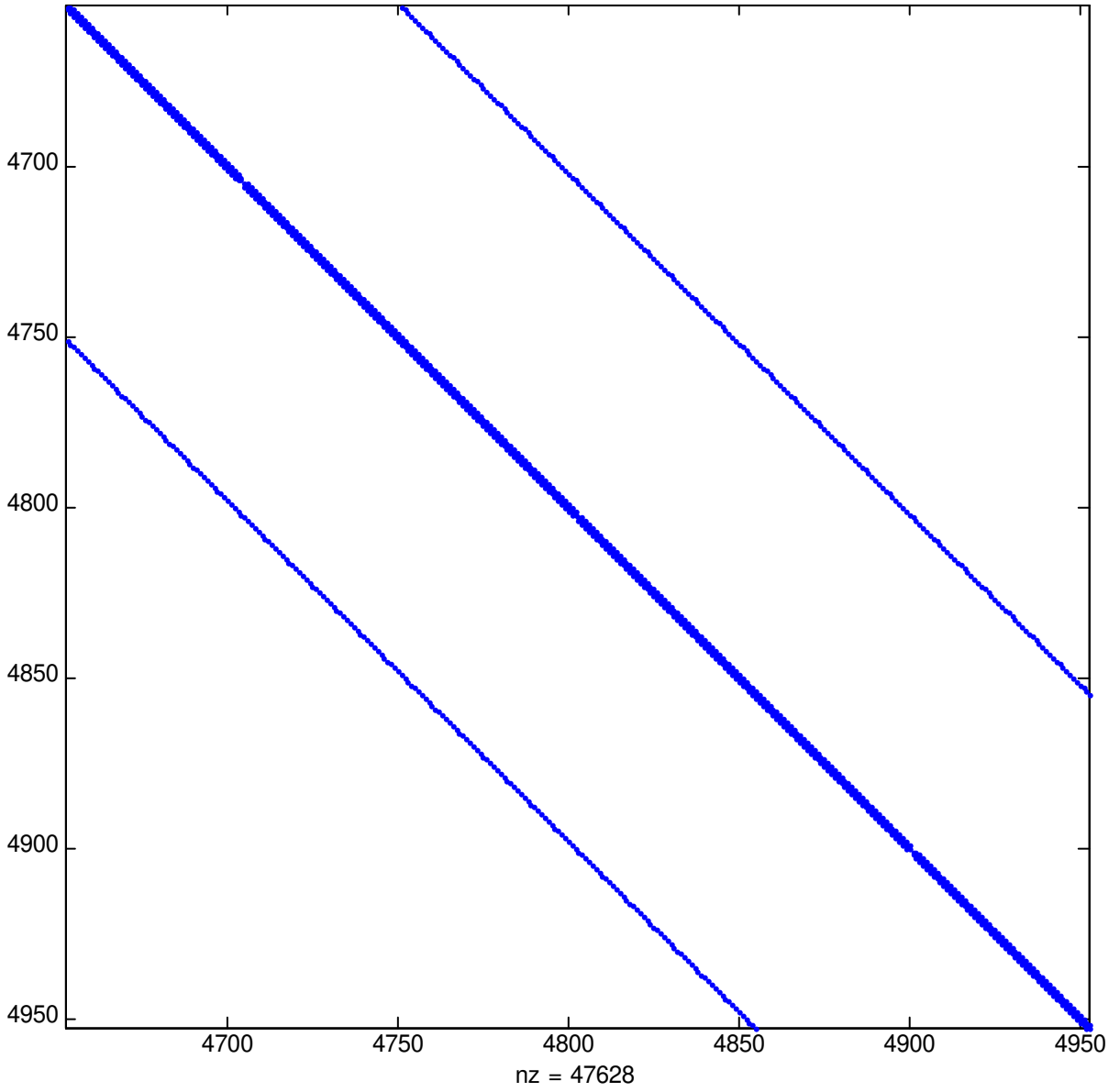




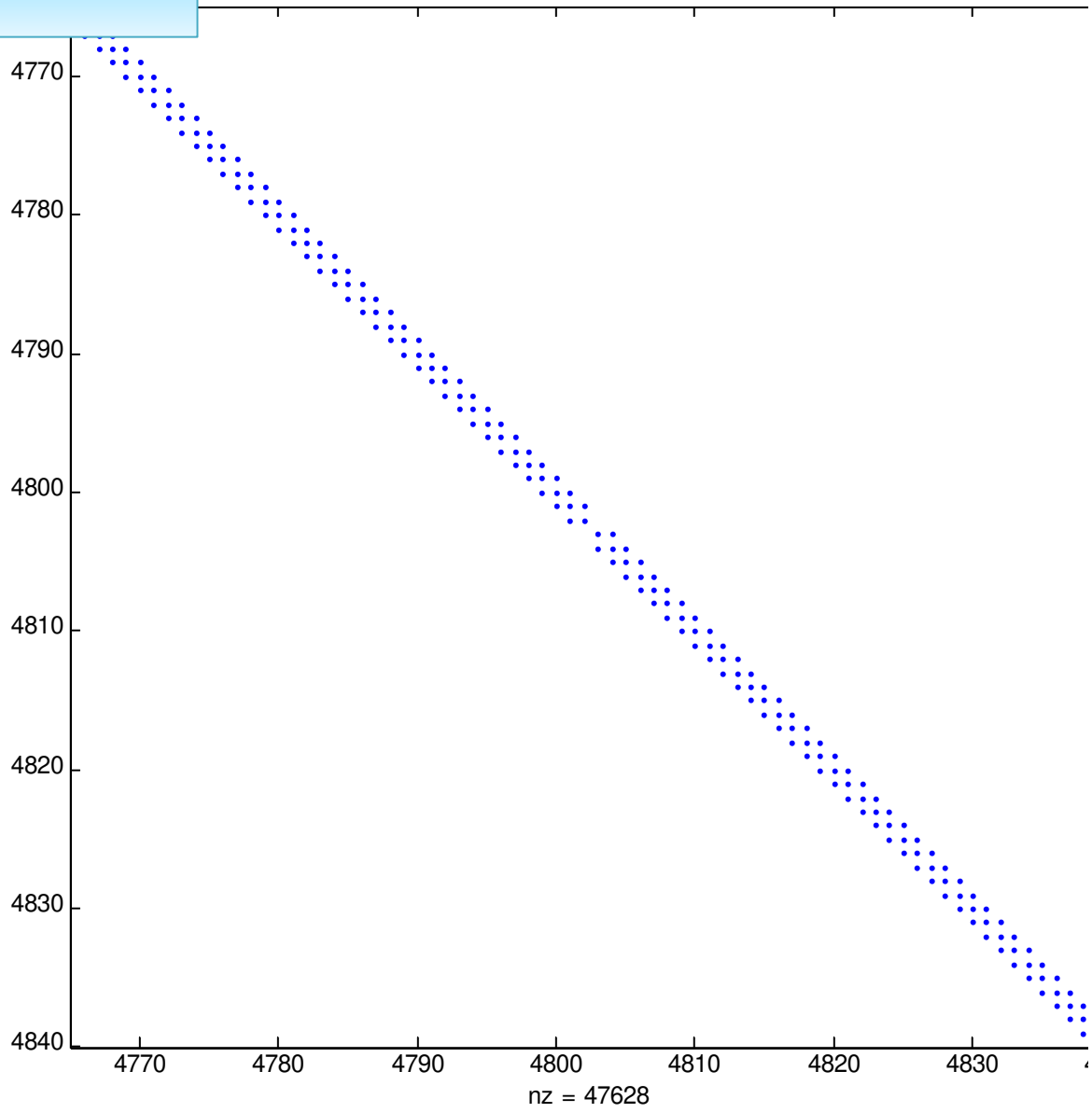
Output of spy(A)



spy(A) after zoom in



Further zoom in of spy(A)



הבנה נוספת: איך מייצגים את הלפלסיאן בבעיה בעזרת הפונקציות המיוחדות במטלב

```
% laplacian demo, 7/12/15:
% laplacian_demo_guy.m
n = 5;
I=speye(n,n)   % sparse eye (identity matrix)
E=sparse(2:n,1:n-1,1,n,n)  % אחד-ים מתחת לאלכסון
D = E+E'-2*I   % מטריצה סימטרית, אחד מעל ומתחת ו-2 באלכסון
A = kron(D,I)+kron(I,D) % see test_kron.m

spy(I)
spy(E)
spy(D)
spy(A)
```

kron

Kronecker tensor product

K = kron(A,B) returns the Kronecker tensor product of matrices A and B. If A is an m-by-n matrix and B is a p-by-q matrix, then kron(A,B) is an m*p-by-n*q matrix formed by taking all possible products between the elements of A and the matrix B.

http://www.mathworks.com/help/matlab/ref/kron.html#zmw57dd0e410809

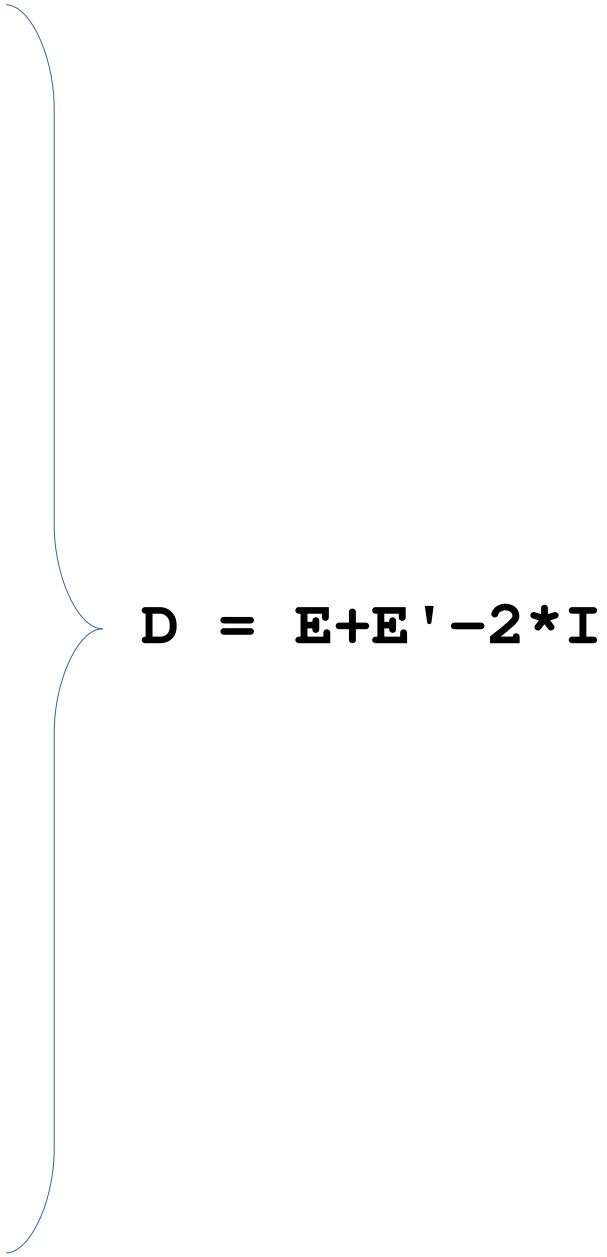
```
>>>I =


Compressed Column Sparse (rows = 5, cols = 5, nnz =
5 [20%])


(1, 1) -> 1

(2, 2) -> 1

(3, 3) -> 1

(4, 4) -> 1

(5, 5) -> 1
```

I=speye(n,n)

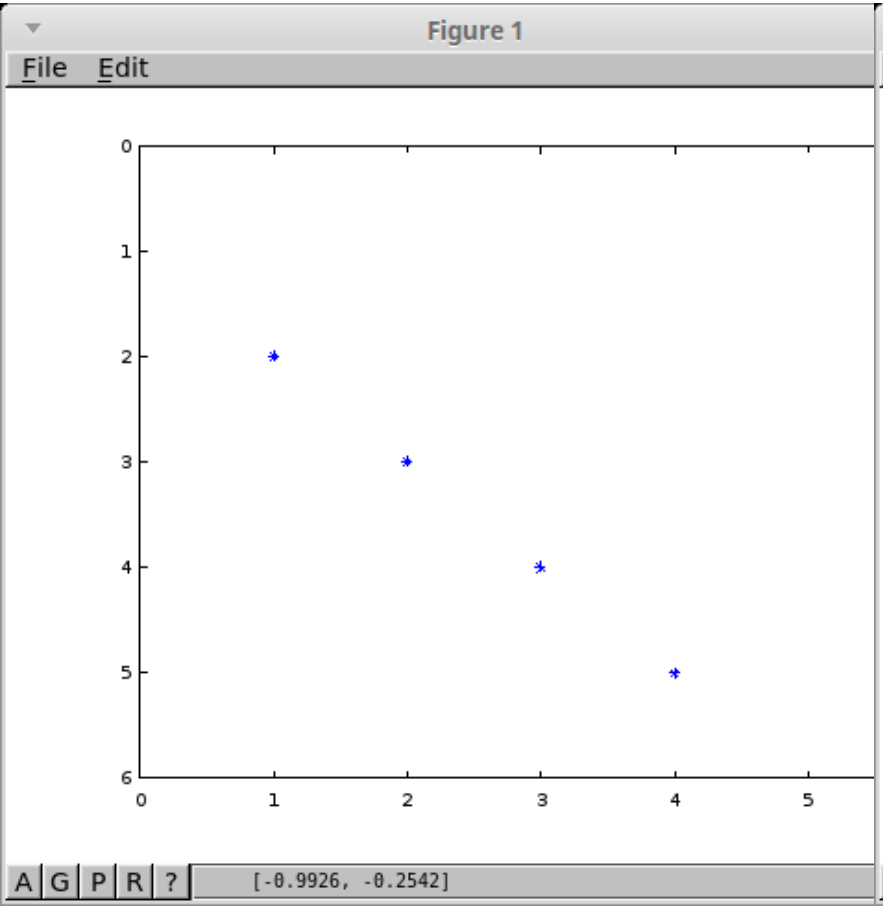
```
E =


Compressed Column Sparse (rows = 5, cols = 5, nnz =
4 [16%])


(2, 1) -> 1

(3, 2) -> 1

(4, 3) -> 1

(5, 4) -> 1
```

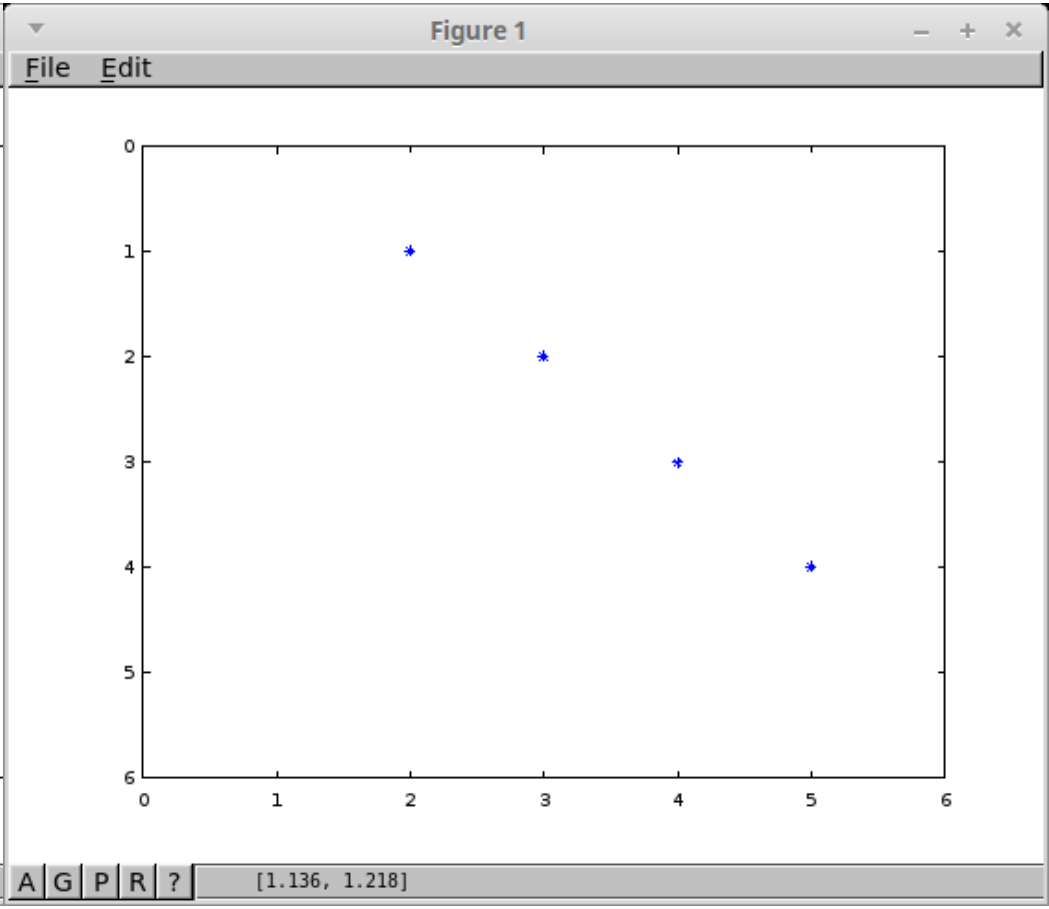
E=sparse(2:n,1:n-1,1,n,n)

```
D =


Compressed Column Sparse (rows =
5, cols = 5, nnz = 13 [52%])


(1, 1) -> -2

(2, 1) -> 1

(1, 2) -> 1

(2, 2) -> -2

(3, 2) -> 1

(2, 3) -> 1

(3, 3) -> -2

(4, 3) -> 1

(3, 4) -> 1

(4, 4) -> -2

(5, 4) -> 1

(4, 5) -> 1

(5, 5) -> -2
```

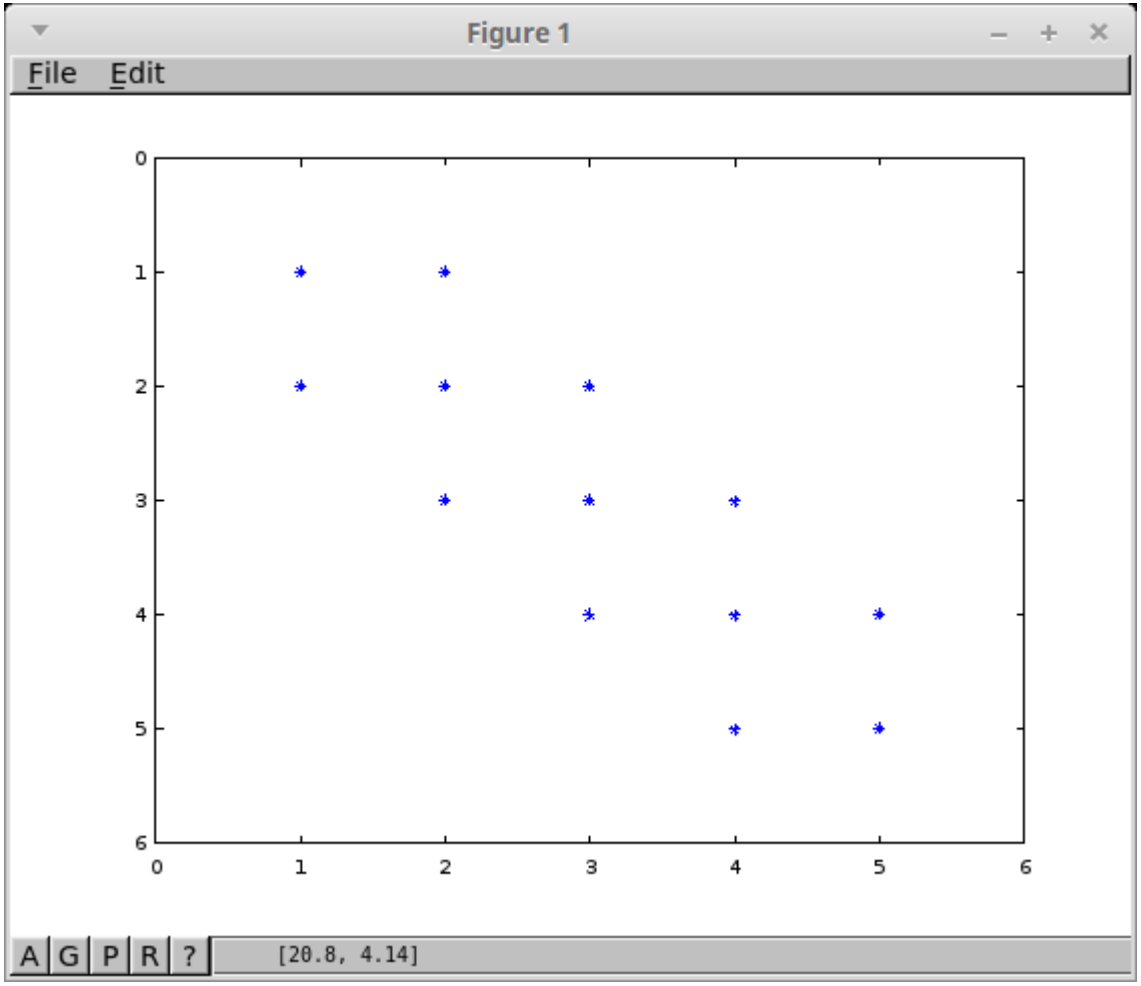


spy(E) – left
spy(E') - right

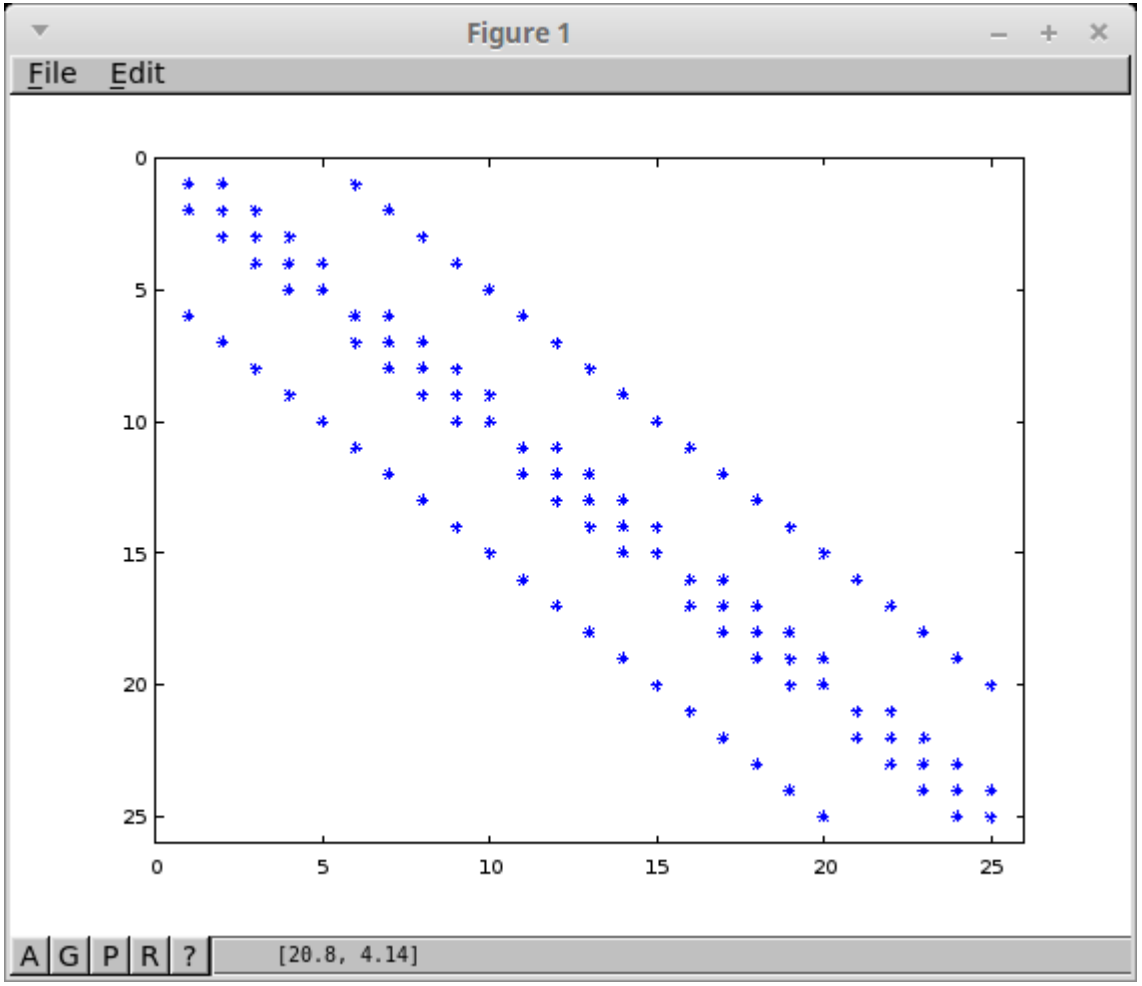




spy(D)



spy(A)



```
>> test_kron

I =

     1     0     0
     0     1     0
     0     0     1


D =

     2     1     0
     1     2     1
     0     1     2


 kron(I,D)

 ans =

      2     1     0     0     0     0     0     0     0
      1     2     1     0     0     0     0     0     0
      0     1     2     0     0     0     0     0     0
      0     0     0     2     1     0     0     0     0
      0     0     0     1     2     1     0     0     0
      0     0     0     0     1     2     0     0     0
      0     0     0     0     0     0     2     1     0
      0     0     0     0     0     0     1     2     1
      0     0     0     0     0     0     0     1     2
```

This slide and the next two demonstrates the effect of kron.
See: test_kron.m

```
>> test_kron

I =

     1     0     0
     0     1     0
     0     0     1


D =

     2     1     0
     1     2     1
     0     1     2
```

```
kron(D,I)

ans =

     2     0     0     1     0     0     0     0     0
     0     2     0     0     1     0     0     0     0
     0     0     2     0     0     1     0     0     0
     1     0     0     2     0     0     1     0     0
     0     1     0     0     2     0     0     1     0
     0     0     1     0     0     2     0     0     1
     0     0     0     1     0     0     2     0     0
     0     0     0     0     1     0     0     2     0
     0     0     0     0     0     1     0     0     2
```

```
sum of both

ans =

     4     1     0     1     0     0     0     0     0
     1     4     1     0     1     0     0     0     0
     0     1     4     0     0     1     0     0     0
     1     0     0     4     1     0     1     0     0
     0     1     0     1     4     1     0     1     0
     0     0     1     0     1     4     0     0     1
     0     0     0     1     0     0     4     1     0
     0     0     0     0     1     0     1     4     1
     0     0     0     0     0     1     0     1     4
```

A simple Jacobi iteration in C and MPI

- Analyze program from:
`http://www.mcs.anl.gov/research/projects/mpi/tutorial/mpiexmpl/src/jacobi/C/main.html`

Demo on my laptop from **BGU-VM** (Vi-HPS VM):
`~/mpi/JACOBI_SIMPLE/jacobi.c`
- Scalasca demo: skin/scan/square of jacobi-scalasca
- jumpshot demo

In this example, we solve the Laplace equation in two dimensions with **finite differences**. This may sound involved, but really amount only to a simple computation, combined with the previous example of a parallel mesh data structure.

Any numerical analysis text will show that iterating

```
while (not converged) {
  for (i,j)
    xnew[i][j] = (x[i+1][j] + x[i-1][j] + x[i][j+1] + x[i][j-1])/4;
  for (i,j)
    x[i][j] = xnew[i][j];
  }
```

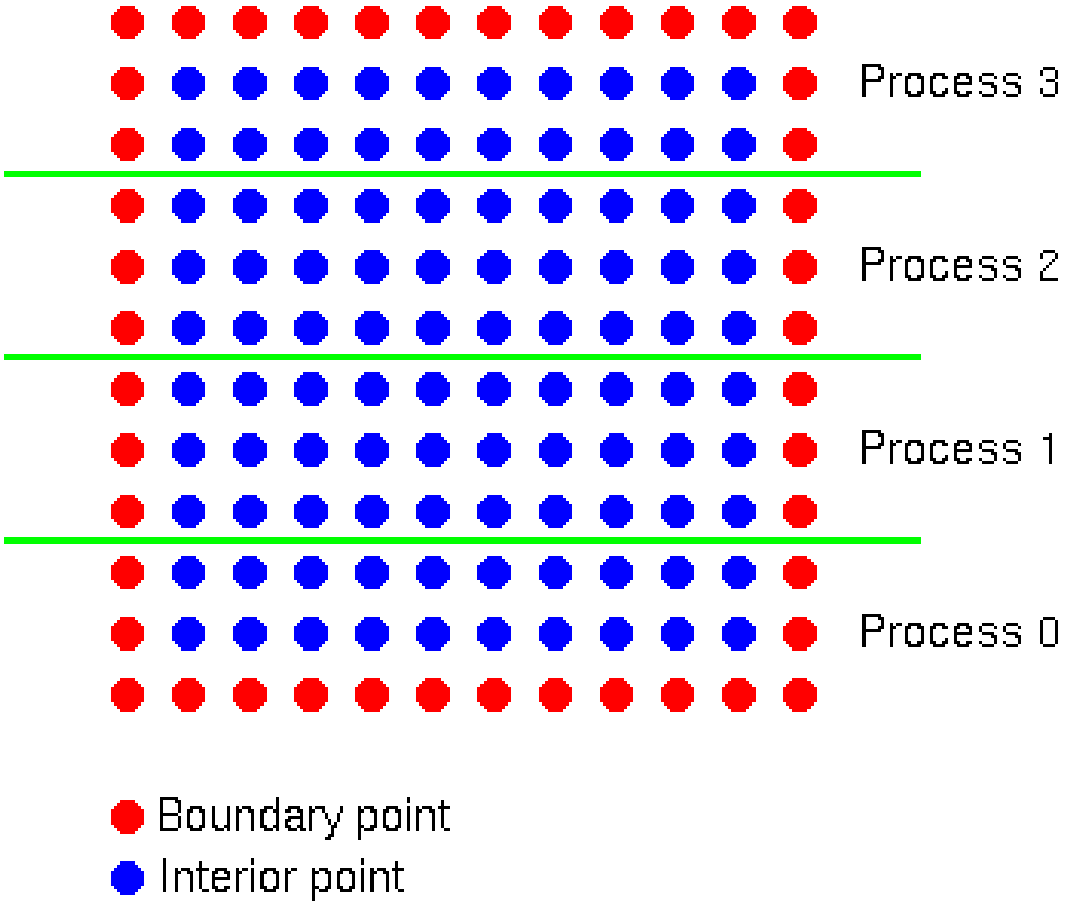
will compute **an approximation** for the solution of **Laplace's equation**.

There is one last detail; this replacement of xnew with the average of the values around it is applied only in the interior; the boundary values are left fixed. In practice, this means that if the mesh is n by n, then the values
x[0][j], x[n-1][j], x[i][0], x[i][n-1]
are left unchanged. Of course, these refer to the complete mesh; you'll have to figure out what to do with for the decomposed data structures (xlocal).
Because the values are replaced by averaging around them, these techniques are called **relaxation methods**.

We wish to compute this approximation in parallel. Write a program to apply this approximation. For convergence testing, compute

```
diffnorm = 0;
for (i,j)
    diffnorm+=(xnew[i][j]-x[i][j]) * (xnew[i][j]-x[i][j]);
diffnorm = sqrt(diffnorm);
```

You'll need to use **`MPI_Allreduce`** for this. (Why not use MPI_Reduce?) Have process zero write out the value of `diffnorm` and the iteration count at each iteration. When `diffnorm` is less that 1.0e-2, consider the iteration converged. Also, if you reach 100 iterations, exit the loop.



For simplicity, consider a 12 x 12 mesh on 4 processors

The example solution uses the following boundary conditions: -1 on the top and bottom, and the rank of the process on the side. The initial data (the values of x that are being relaxed) are also the same; the interior points have the same value as the rank of the process. This is shown below:

```
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 3  3  3  3  3  3  3  3  3  3  3  3
 3  3  3  3  3  3  3  3  3  3  3  3
 2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2
 1  1  1  1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1  1  1  1
 0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

2D Jacobi iterations

In the next 5 slides the c program is given including an example how to add your own MPE(*) defined events to be watched later in Jumpshot

(*)=Emphasized in **bold**

MPE home page:
https://www.mcs.anl.gov/research/projects/perfvis/software/MPE/

```
#include <stdio.h>
#include <math.h>
#include "mpi.h"
#include "mpe.h"

/* This example handles a 12 x 12 mesh, on 4 processors only. */
#define maxn 12

int main( argc, argv )
int argc;
char **argv;
{
    int        rank, value, size, errcnt, toterr, i, j, itcnt;
    int        i_first, i_last;
    MPI_Status status;
    double     diffnorm, gdiffnorm;
    double     xlocal[(12/4)+2][12];
//  double     xnew[(12/3)+2][12];
    double     xnew[(12/4)+2][12];
    int event1a, event1b, event2a, event2b,
        event3a, event3b, event4a, event4b;
    MPI_Init( &argc, &argv );

    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
```

```
    if (size != 4) MPI_Abort( MPI_COMM_WORLD, 1 );

   // MPE block inserted by Guy:

    MPE_Init_log();
    MPE_Log_get_state_eventIDs( &event1a, &event1b );
    MPE_Log_get_state_eventIDs( &event2a, &event2b );
    MPE_Log_get_state_eventIDs( &event3a, &event3b );
    MPE_Log_get_state_eventIDs( &event4a, &event4b );

    if ( rank == 0 ) {
        MPE_Describe_state( event1a, event1b, "Exchange", "red" );
        MPE_Describe_state( event2a, event2b, "Send", "orange" );
        MPE_Describe_state( event3a, event3b, "Recv", "blue" );
        MPE_Describe_state( event4a, event4b, "AllReduce", "green" );
    }
// End MPE block



    /* xlocal[][0] is lower ghostpoints, xlocal[][maxn+2] is upper */
```

```
    /* Note that top and bottom processes have one less row of
interior points */
    i_first = 1;
    i_last  = maxn/size;
    if (rank == 0)        i_first++;
    if (rank == size - 1) i_last--;

    /* Fill the data as specified */
    for (i=1; i<=maxn/size; i++)
       for (j=0; j<maxn; j++)
          xlocal[i][j] = rank;
    for (j=0; j<maxn; j++) {
       xlocal[i_first-1][j] = -1;
       xlocal[i_last+1][j]  = -1;
    }

    itcnt = 0;
    do {
    /* Send up unless I'm at the top, then receive from below */
    /* Note the use of xlocal[i] for &xlocal[i][0] */
```

```
        MPE_Log_event( event1a, 0, NULL );
    if (rank < size - 1) { // I am not at the top
            MPE_Log_event( event2a, 0, NULL );
        MPI_Send( xlocal[maxn/size], maxn, MPI_DOUBLE, rank + 1, 0,
                  MPI_COMM_WORLD );
            MPE_Log_event( event2b, 0, NULL ); }
    if (rank > 0) { // I am not at the bottom
            MPE_Log_event( event3a, 0, NULL );
        MPI_Recv( xlocal[0], maxn, MPI_DOUBLE, rank - 1, 0,
                  MPI_COMM_WORLD, &status );
           MPE_Log_event( event3b, 0, NULL ); }
    /* Send down unless I'm at the bottom */
    if (rank > 0) {
            MPE_Log_event( event2a, 0, NULL );
        MPI_Send( xlocal[1], maxn, MPI_DOUBLE, rank - 1, 1,
                  MPI_COMM_WORLD );
            MPE_Log_event( event2b, 0, NULL ); }
    if (rank < size - 1) {
            MPE_Log_event( event3a, 0, NULL );
        MPI_Recv( xlocal[maxn/size+1], maxn, MPI_DOUBLE, rank + 1, 1,
                  MPI_COMM_WORLD, &status );
            MPE_Log_event( event3b, 0, NULL ); }
    MPE_Log_event( event1b, 0, NULL );
/* Compute new values (but not on boundary) */
    itcnt ++;
    diffnorm = 0.0;
```

```
    for (i=i_first; i<=i_last; i++)
        for (j=1; j<maxn-1; j++) {
            xnew[i][j] = (xlocal[i][j+1] + xlocal[i][j-1] +
                  xlocal[i+1][j] + xlocal[i-1][j]) / 4.0;
        diffnorm += (xnew[i][j] - xlocal[i][j]) *
                    (xnew[i][j] - xlocal[i][j]);
        }
    /* Only transfer the interior points */
    for (i=i_first; i<=i_last; i++)
        for (j=1; j<maxn-1; j++)
        xlocal[i][j] = xnew[i][j];
        MPE_Log_event( event4a, 0, NULL );
    MPI_Allreduce( &diffnorm, &gdiffnorm, 1, MPI_DOUBLE, MPI_SUM,
                MPI_COMM_WORLD );
        MPE_Log_event( event4b, 0, NULL );
    gdiffnorm = sqrt( gdiffnorm );
    if (rank == 0) printf( "At iteration %d, diff is %e\n", itcnt,
                   gdiffnorm );
    } while (gdiffnorm > 1.0e-2 && itcnt < 100);

    MPE_Log_sync_clocks();
    MPE_Finish_log( "jacobilog" );
    MPI_Finalize( );
    return 0;
}
```

The Makefile

```
# Guy Tel-Zur, 1/5/2016,
#/home/telzur/Documents/Teaching/PP20XXX/lectures/07/code/JACOBI_SIMP
LE
SHELL = /bin/sh
jacobi: jacobi.c
    mpicc -o jacobi jacobi.c -lm
jacobi_mpe: jacobi.c
    mpecc -o jacobi_mpe jacobi.c -mpilog -lm -lpthread
#----including the exchange profiling:
jacobi_mpe_guy: jacobi.c
    mpecc -o jacobi_mpe_guy jacobi_mpe.c -mpilog -lm -lpthread

jacobi_scalasca: jacobi.c
    skin mpicc -o jacobi_scalasca jacobi.c -lm
clean:
    /bin/rm -f jacobi jacobi.o jacobi.log
```

Profiling with Jumpshot

Hobbits:

```
mpicc  -o jacobi_mpe ./jacobi_mpe.c \
-lm  -lmpe -llmpe
```

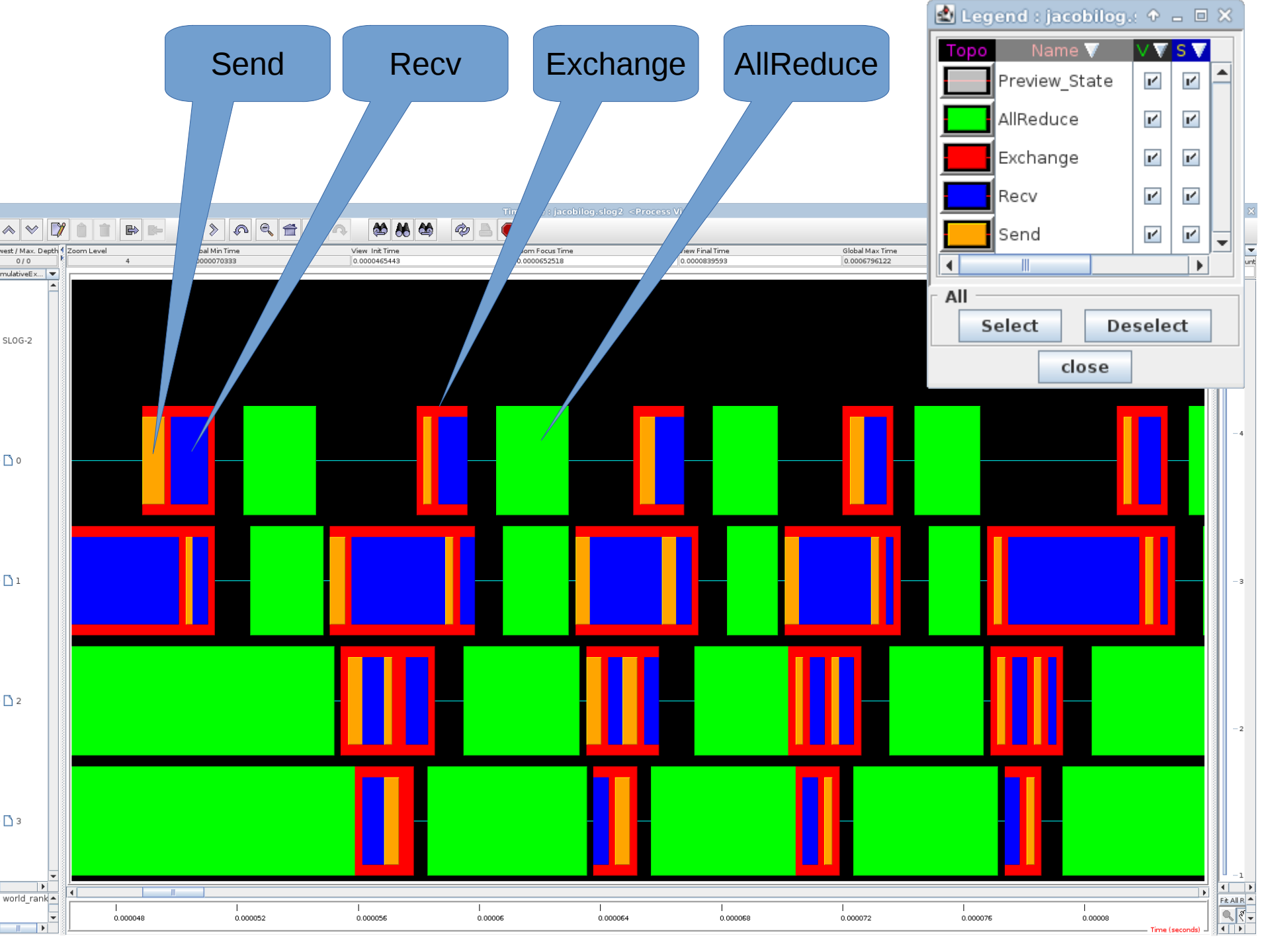
OR:

```
mpecc  -o jacobi_mpe ./jacobi_mpe.c \
-mpilog -lm [-lpthread]

clog2TOslog2 ./jacobilog.clog2
jumpshot jacobilog.slog2
```

In BGU-VM (Vi-HPS): Generate jumpshot files using TAU (tau_cc.sh)

```
mpirun -np 4/jacobi_mpe

Look at the source code to see
additional MPE events calls
```

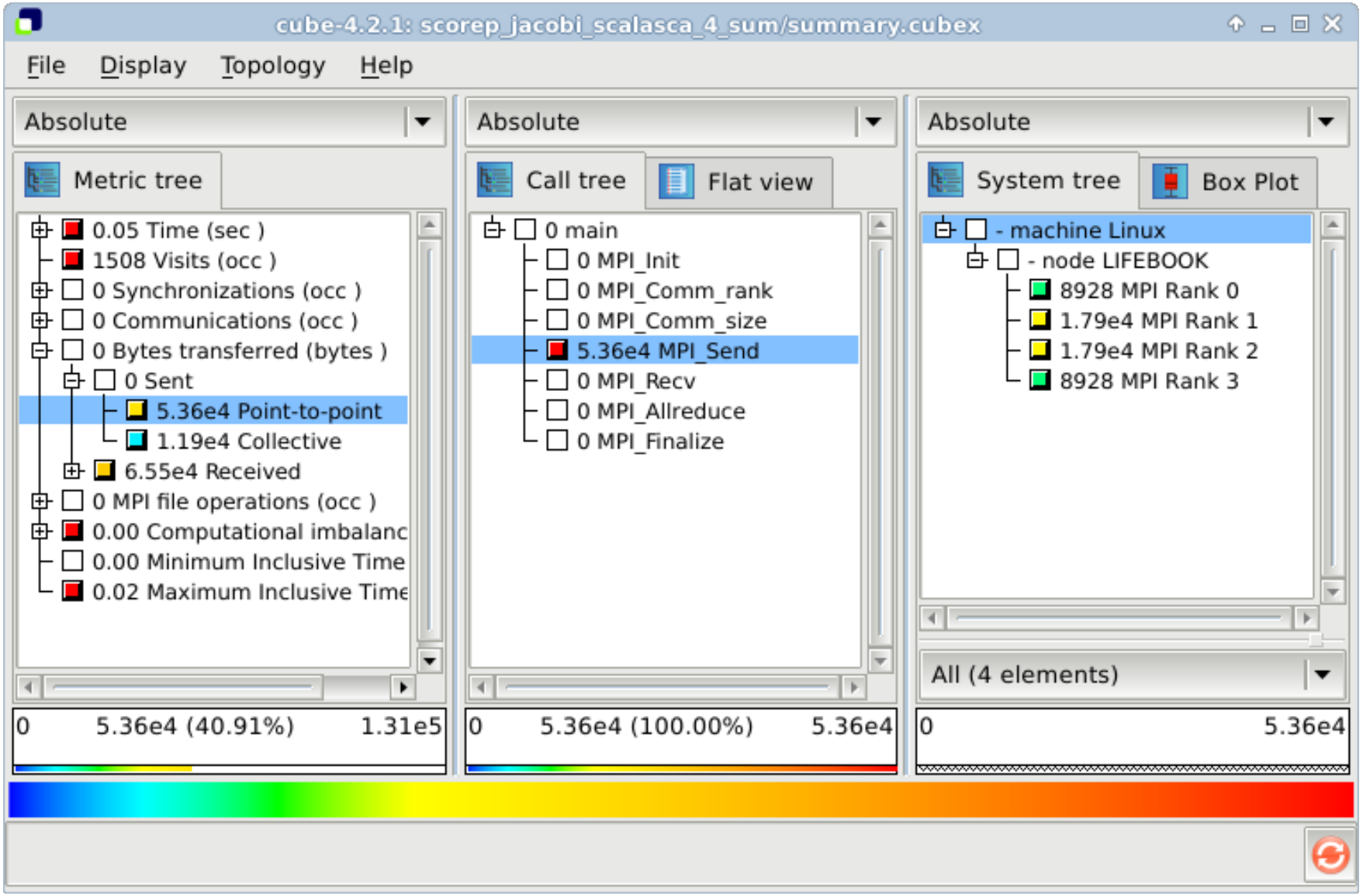
Jacobi demo also on vihps under:~/07/code/JACOBI


Send
Recv
Exchange
AllReduce
Legend : jacobilog.
Topo
Name
V
S
Preview_State
AllReduce
Exchange
Recv
Send
All
Select
Deselect
close
SLOG-2
0
1
2
3
world_rank
0.000048
0.000052
0.000056
0.00006
0.000064
0.000068
0.000072
0.000076
0.00008
Time (seconds)

Profiling with Scalasca

שורת המשך

```
scalasca -instrument mpicc -o  \
jacobi_scalasca ./jacobi.c -lm

scalasca -analyze mpirun -np 4  \
./jacobi_scalasca


scalasca -examine \
./scorep_jacobi_scalasca_4_sum
```


cube-4.2.1: scorep_jacobi_scalasca_4_sum/summary.cubex
File
Display
Topology
Help
Absolute
Metric tree
0.05 Time (sec)
1508 Visits (occ)
0 Synchronizations (occ)
0 Communications (occ)
0 Bytes transferred (bytes)
0 Sent
5.36e4 Point-to-point
1.19e4 Collective
6.55e4 Received
0 MPI file operations (occ)
0.00 Computational imbalanc
0.00 Minimum Inclusive Time
0.02 Maximum Inclusive Time
0
5.36e4 (40.91%)
1.31e5
Absolute
Call tree
Flat view
0 main
0 MPI_Init
0 MPI_Comm_rank
0 MPI_Comm_size
5.36e4 MPI_Send
0 MPI_Recv
0 MPI_Allreduce
0 MPI_Finalize
0
5.36e4 (100.00%)
5.36e4
Absolute
System tree
Box Plot
- machine Linux
- node LIFEBOOK
8928 MPI Rank 0
1.79e4 MPI Rank 1
1.79e4 MPI Rank 2
8928 MPI Rank 3
All (4 elements)
0
5.36e4

MPI Virtual Topologies

Cartesian Constructor

Makes a new communicator to which topology information has been attached

```
int MPI_Cart_create( MPI_Comm comm_old, int ndims, int *dims,
int *periods, int reorder, MPI_Comm *comm_cart )
```

Parameter	Meaning of Parameter
comm_old	input communicator (handle)
ndims	number of dimensions of cartesian grid (integer)
dims	integer array of size ndims specifying the number of processes in each dimension
periods	logical array of size ndims specifying whether the grid is periodic (true) or not (false) in each dimension
reorder	ranking may be reordered (true) or not (false) (logical)
comm_cart	communicator with new cartesian topology (handle)

Example 1

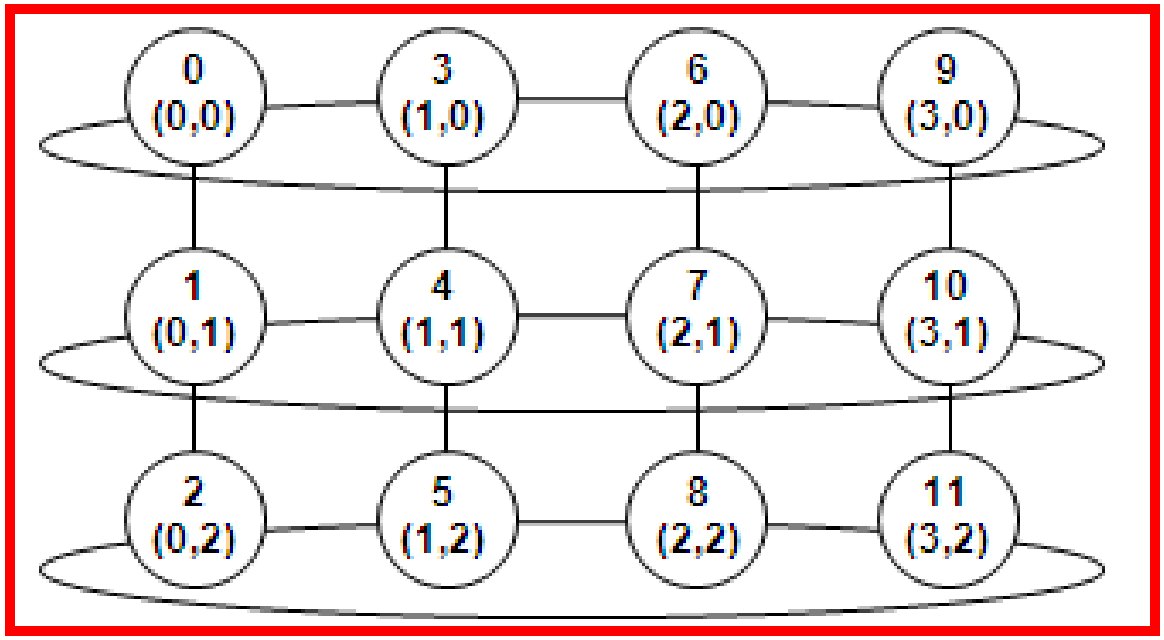
Torus in one direction.

comm_old = MPI_COMM_WORLD

ndims = 2

dims = (4, 3)

periods = (1/.true., 0/.false.)



MPI_Cart_shift

Returns the shifted source and destination ranks, given a shift direction and amount

Synopsis

int MPI_Cart_shift(MPI_Comm comm, int direction, int displ, int *source,
int *dest)

Input Parameters:

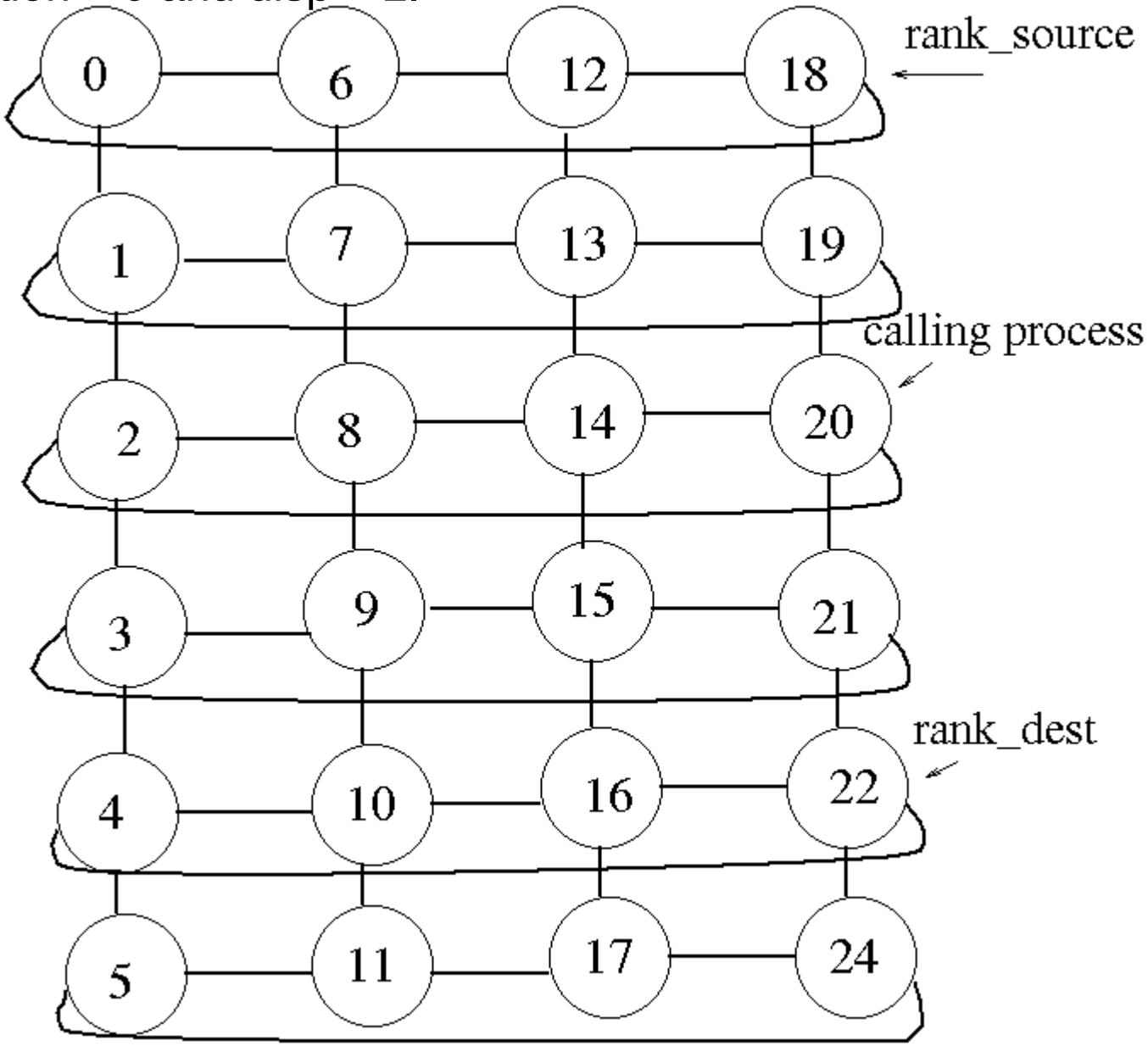
comm - communicator with cartesian structure (handle)
direction - coordinate dimension of shift (integer)
displ - displacement (> 0: upwards shift, < 0: downwards shift) (integer)

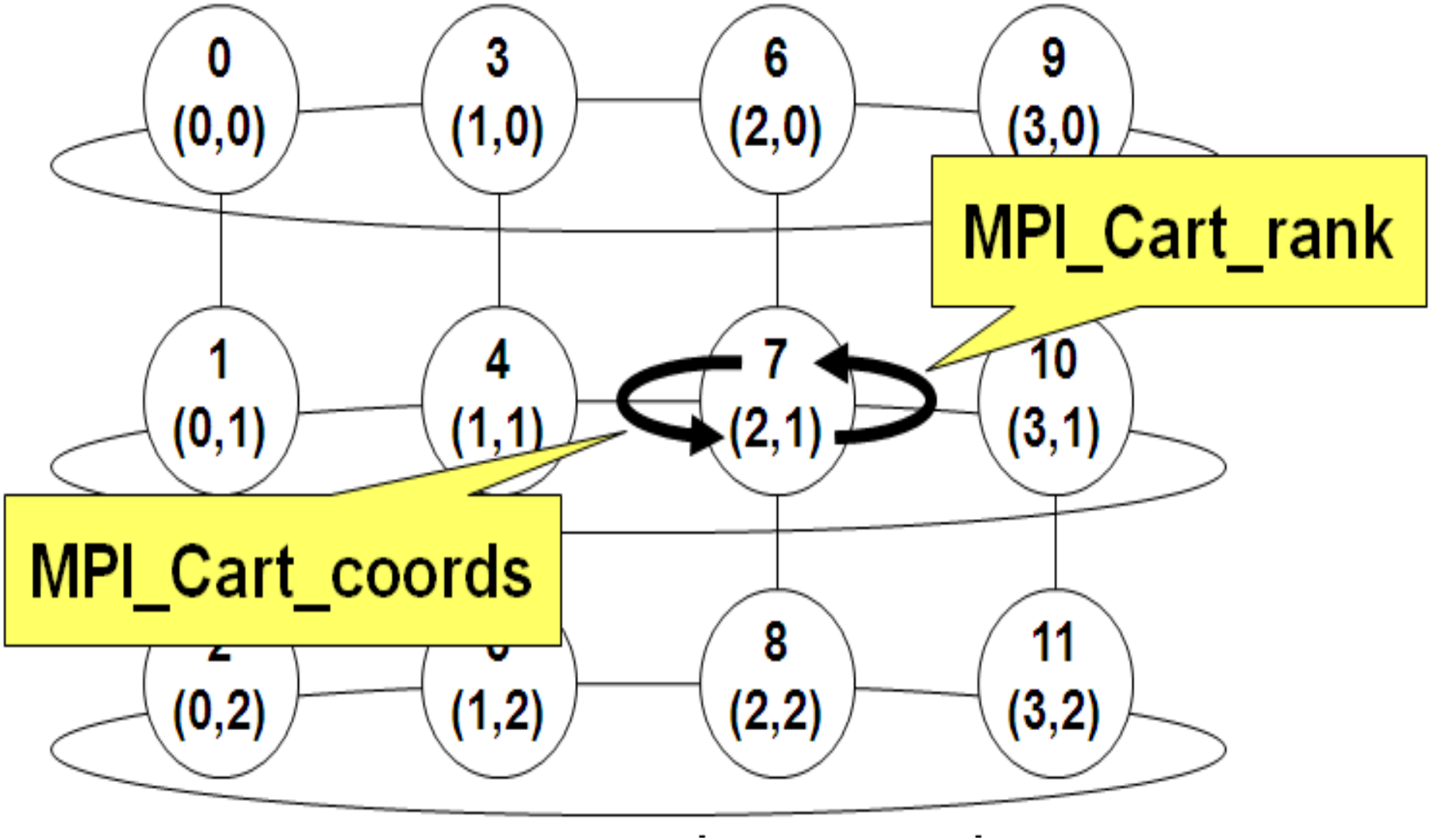
Output Parameters

source - rank of source process (integer)
dest - rank of destination process (integer)

http://www.mcs.anl.gov/research/projects/mpi/www/www3/MPI_Cart_shift.html

The following virtual topology shows MPI_cart_shift called on process 20 with direction = 0 and disp = 2.




0
(0,0)
3
(1,0)
6
(2,0)
9
(3,0)
MPI_Cart_rank
1
(0,1)
4
(1,1)
7
(2,1)
10
(3,1)
MPI_Cart_coords
8
(2,2)
11
(3,2)

Jacobi Iteration using MPI virtual topologies

Reference: Intermediate MPI from "***Using MPI***" book, a FORTRAN example:

http://www.mcs.anl.gov/research/projects/mpi/usingmpi/examples-usingmpi/intermediate/index.html

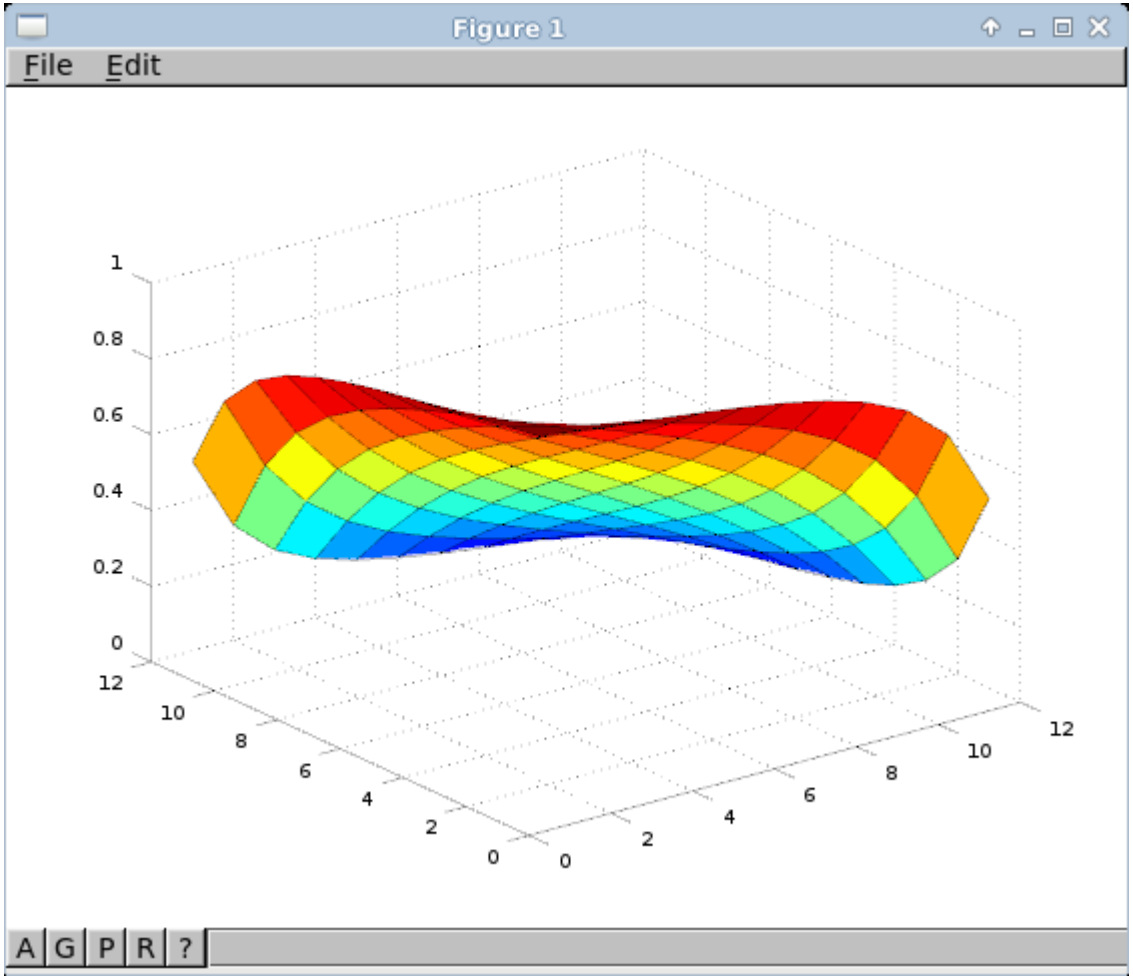
oned.c – A C program example

Teacher's note:
Show and analyze the code from the browser then do the demo.
Demo: …/07/code/JACOBI_INTERMEDIATE (also show on BGU=VM (Vi-HPS)).
See Makefile for regular built and Scalasca built

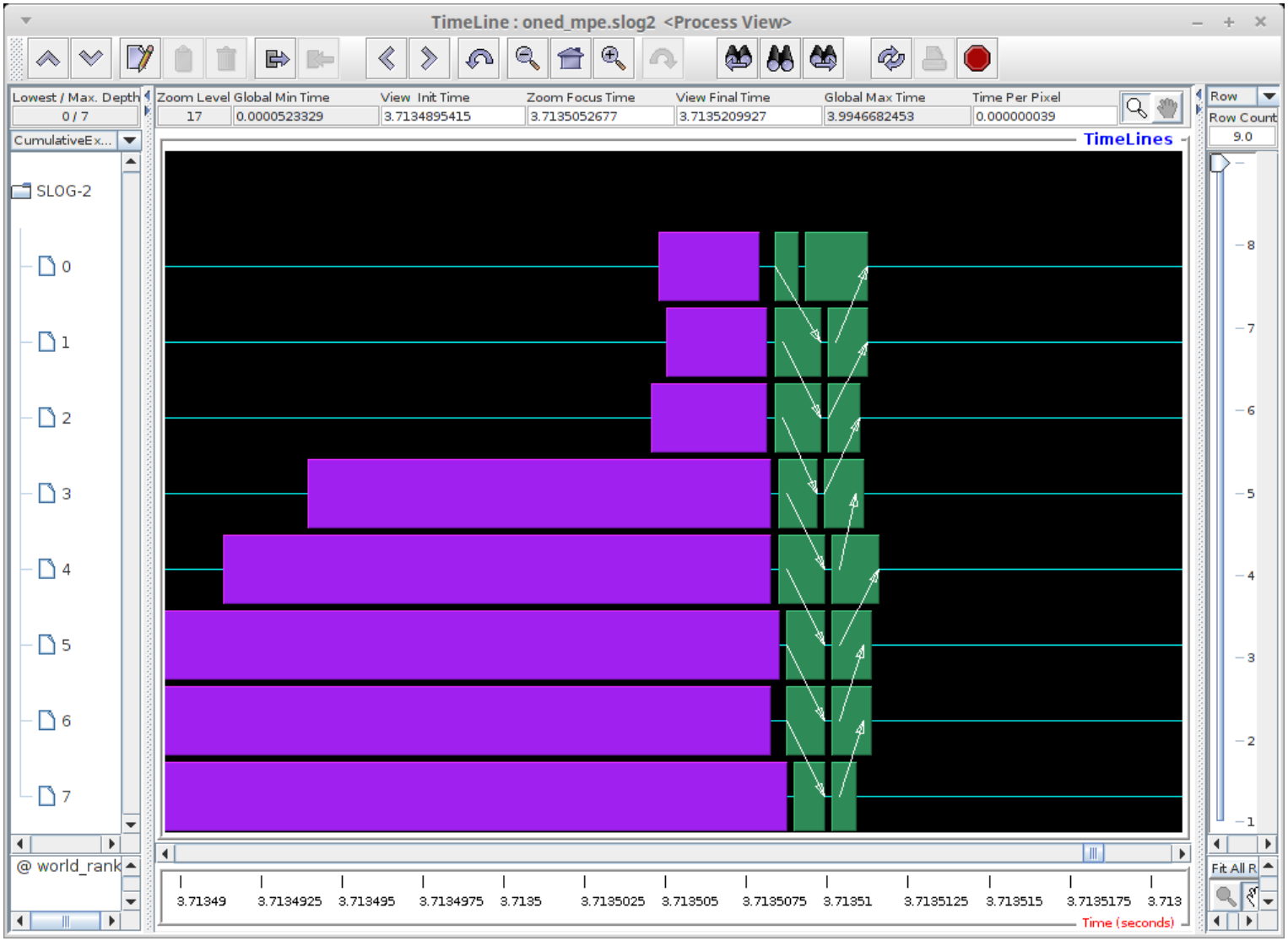
```
telzur $ mpirun -np 4 ./oned
Enter nx: 12
Task: 0, column 1 to 3
Task: 1, column 4 to 6
Task: 2, column 7 to 9
Task: 3, column 10 to 12
0 its. Difference is 0.468750
100 its. Difference is 0.000060

Converged after 172 iterations
172 its. Difference is 0.000001
CPU time (s): 0.000708
Real time (s): 0.000713
```

```
>>> load("oned.cout")
>>> surf(oned)
```

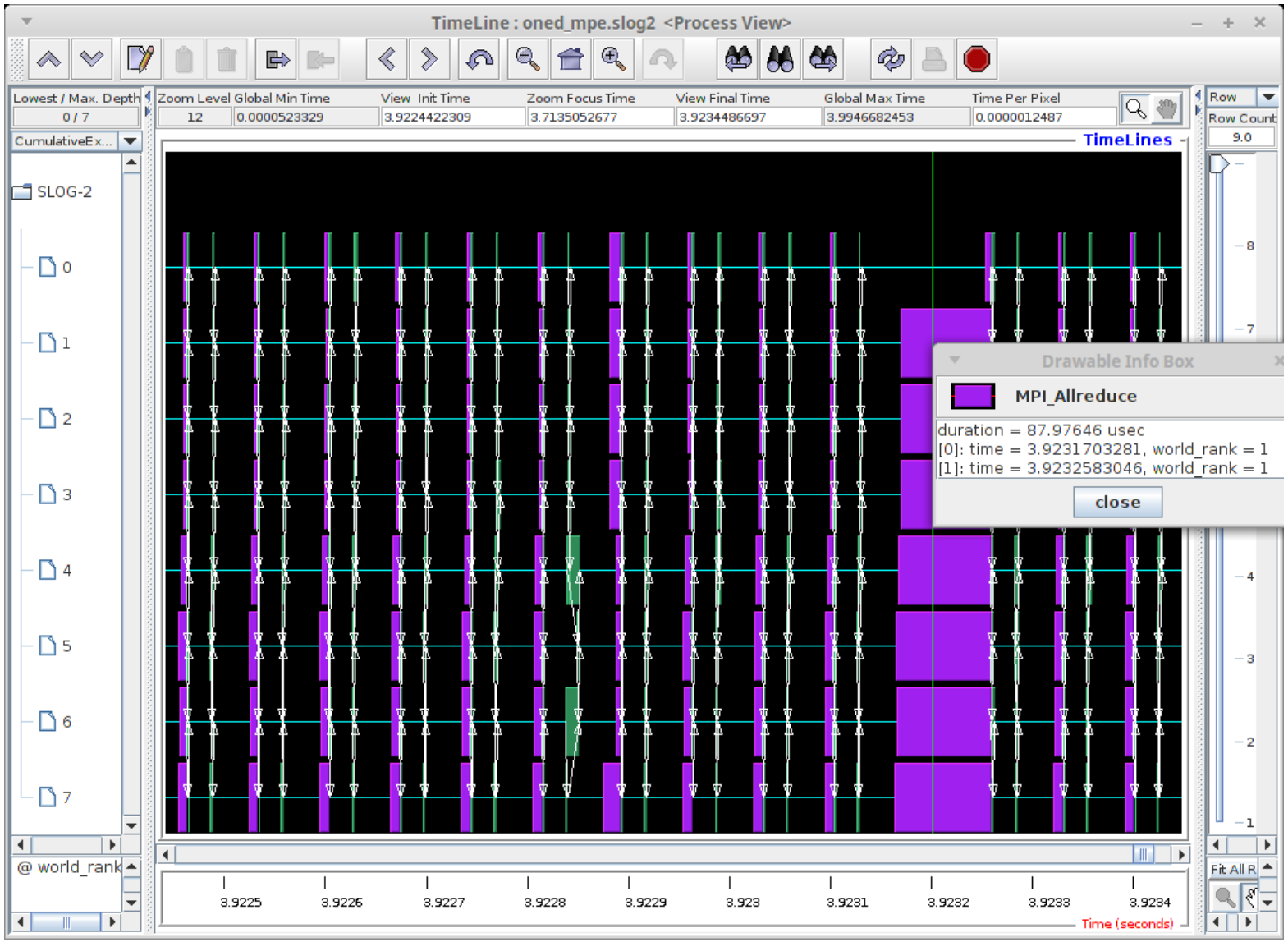
In Matlab



Check the tracing in Jumpshot and notice the "**exchange**" between nodes (local synchronization)

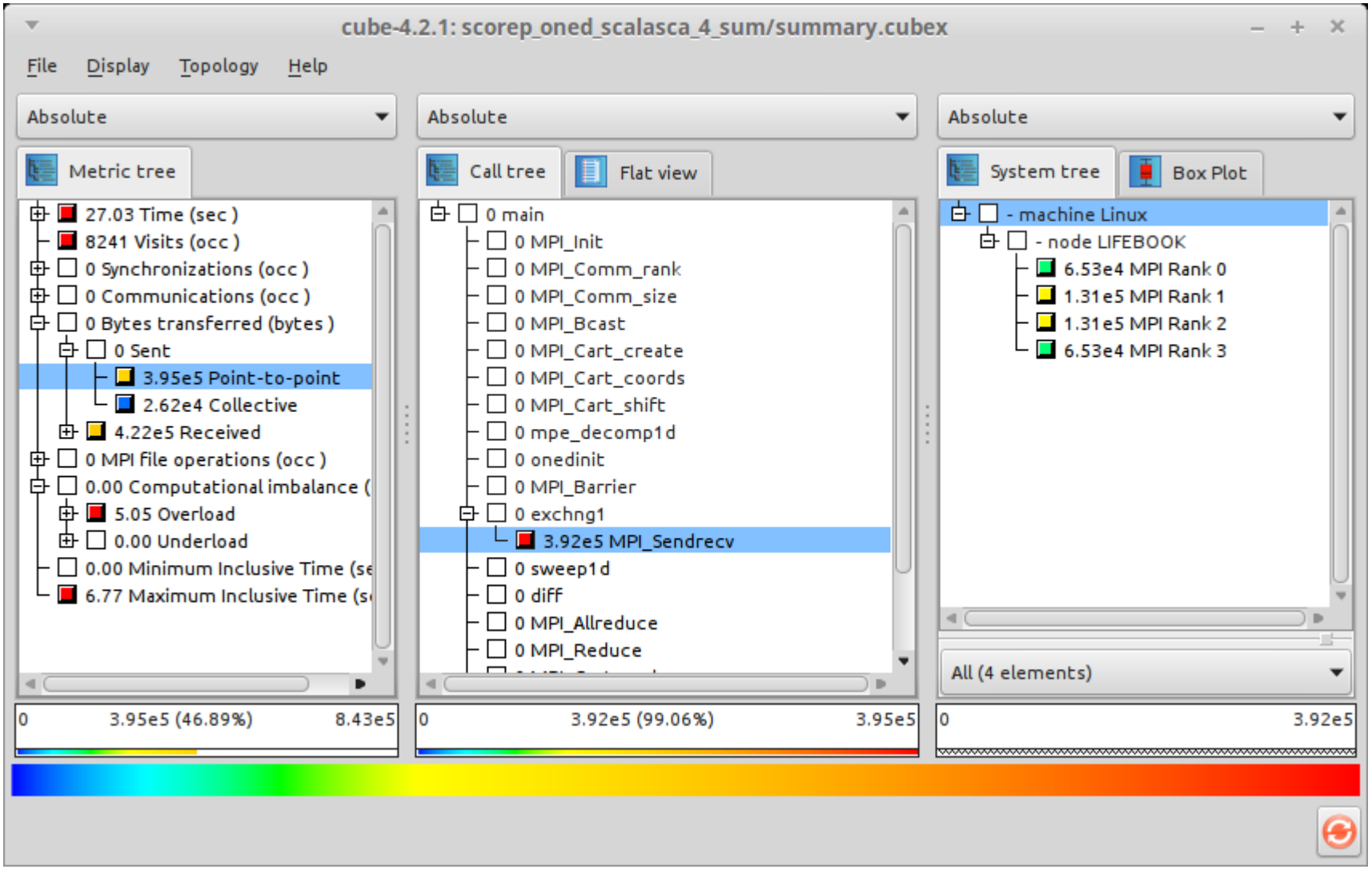


oned.c: Allreduce

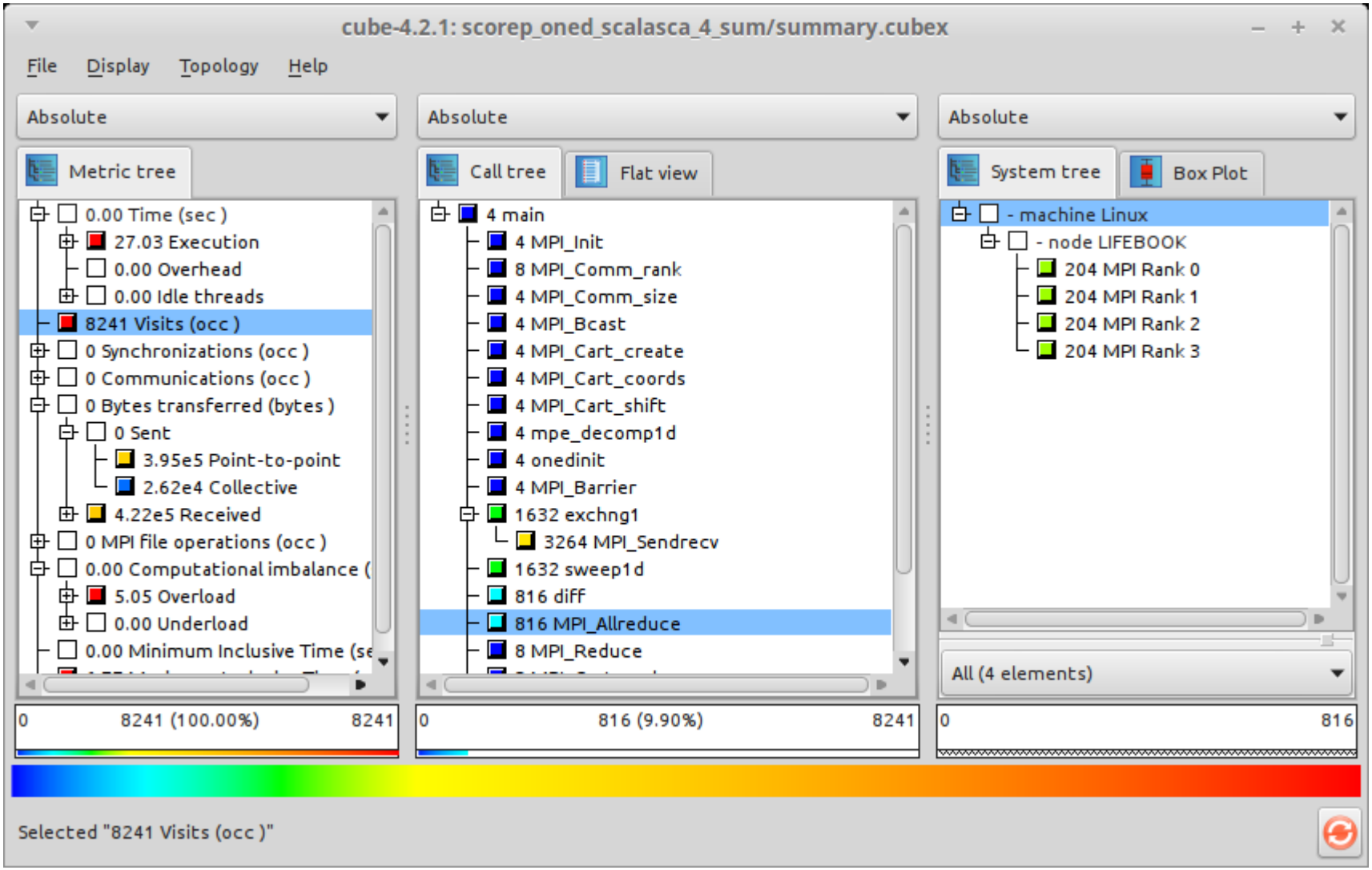


message volume (bytes)
receiver
sender
message volume (bytes)

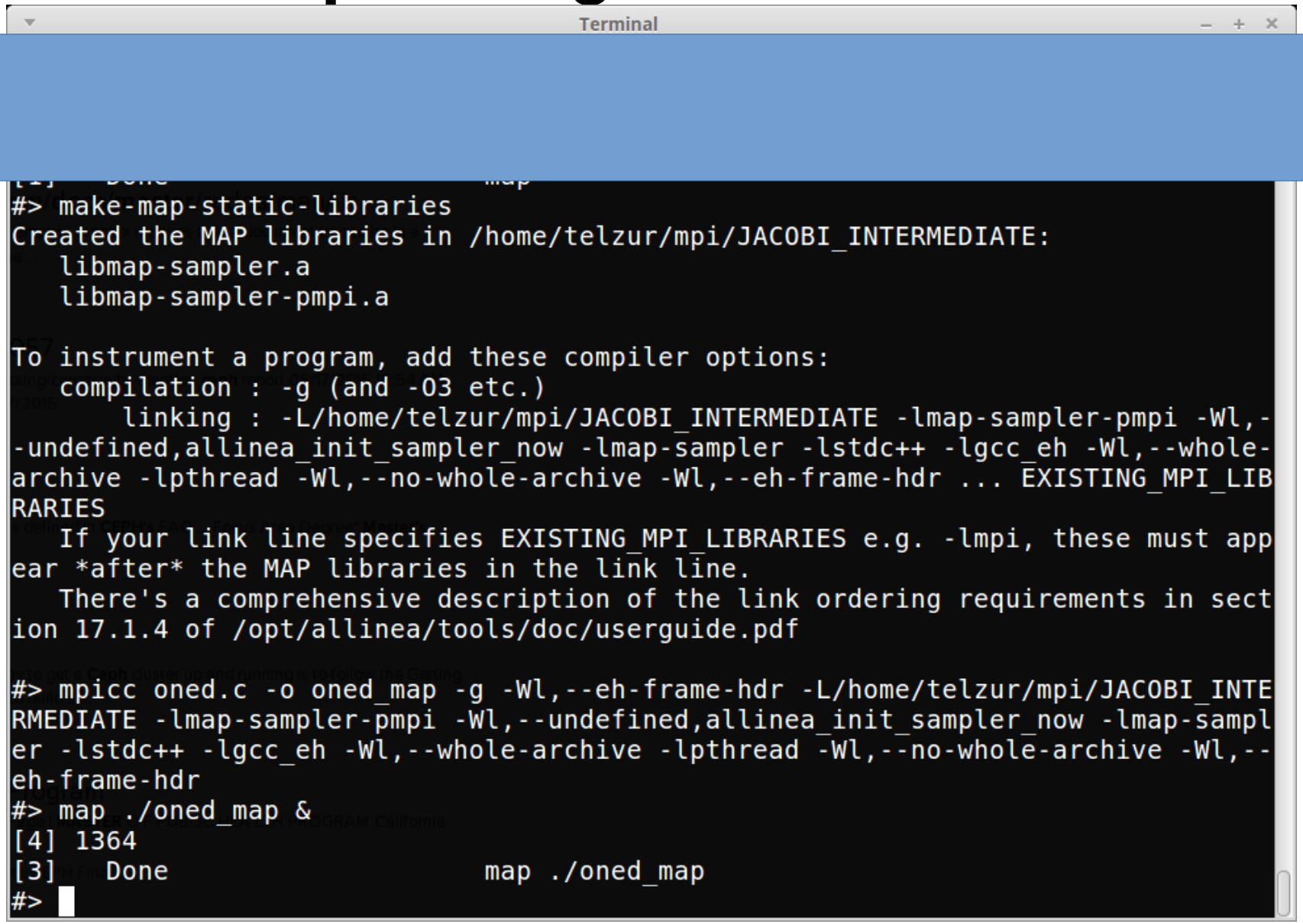
oned.c: Scalasca



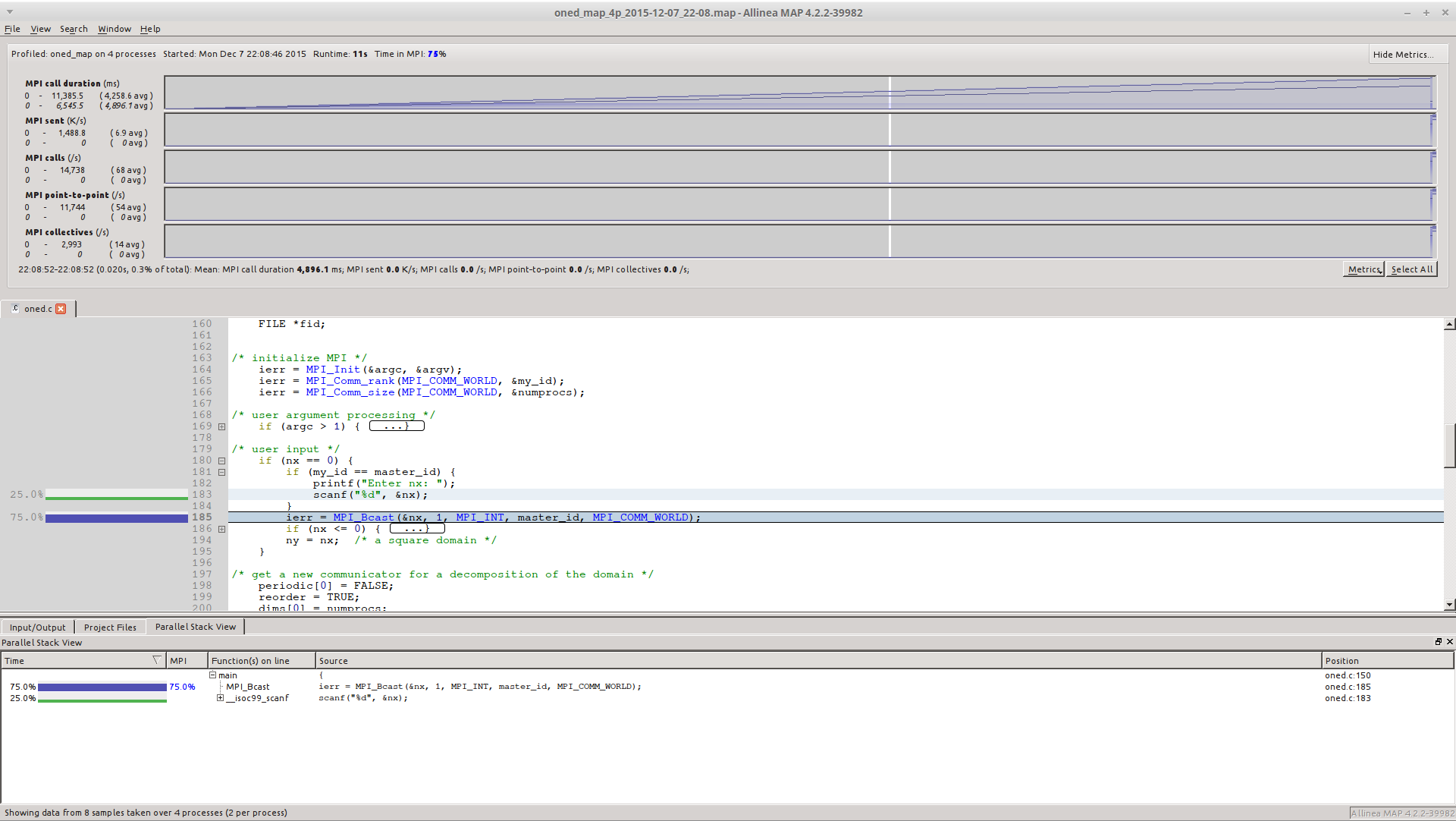
oned.c: Scalasca



Oned.c profiling with Allinea MAP



Oned.c profiling with Allinea MAP



Time dependent 1D wave equation

Ref: Solving Problems on Concurrent Processors, Volume 1 by **G. Fox** et. al. Chapter 5

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u$$

Lecture07/code/WAVE/wave.c

HPC wire

Since 1987 - Covering the Fastest Computers in the World and the People Who Run Them

Geoffrey C. Fox Named Recipient of 2019 ACM-IEEE CS Ken Kennedy Award

October 23, 2019

NEW YORK, October 23, 2019 – The Association for Computing Machinery (ACM) and IEEE Computer Society (IEEE-CS) have named Geoffrey C. Fox of Indiana University Bloomington as the recipient of the 2019 ACM-IEEE CS Ken Kennedy Award. Fox was cited for foundational contributions to parallel computing methodology, algorithms and software, and data analysis, and their interfaces with broad classes of applications. The award will be presented at SC 19: The International Conference for High Performance Computing, Networking, Storage and Analysis, November 17-22, in Denver, Colorado.

Geoffrey Fox. Image courtesy of ACM.

Through a long and distinguished career, Fox has made several important technical contributions to high performance computing. Fox identified the principles behind the use of decomposition and efficient message passing in early MIMD (multiple instruction, multiple data) hypercubes., which pioneered application development on parallel machines. In several well-received papers, Fox demonstrated the synergies between Message Passing Interface (MPI) and MapReduce. In one paper, for instance, he introduced the programming model and architecture of Twister, an enhanced map-reduce runtime that supports iterative MapReduce computations efficiently. His more recent Twister 2 system systematically provides HPC performance with functionalities similar to Apache Spark, Flink, Storm, and Heron. His recent work at the interface of HPC and data-intensive computing has resulted in the SPIDAL (Scalable Parallel and Interoperable Data-intensive Application Library) project. SPIDAL supports a very diverse collection of data-intensive applications on high performance computing platforms.

program for the numerical solution of this equation will be to propose a uniform
etization of the spatial and temporal domains, and approximate the partial differen-
equation of Eq. (5-1) by a finite difference expression. If Δx and Δt represent the
e and time step sizes respectively, and ψ_i represents the approximate solution at the
odal point in the spatial discretization, then the second order finite difference equa-
becomes:

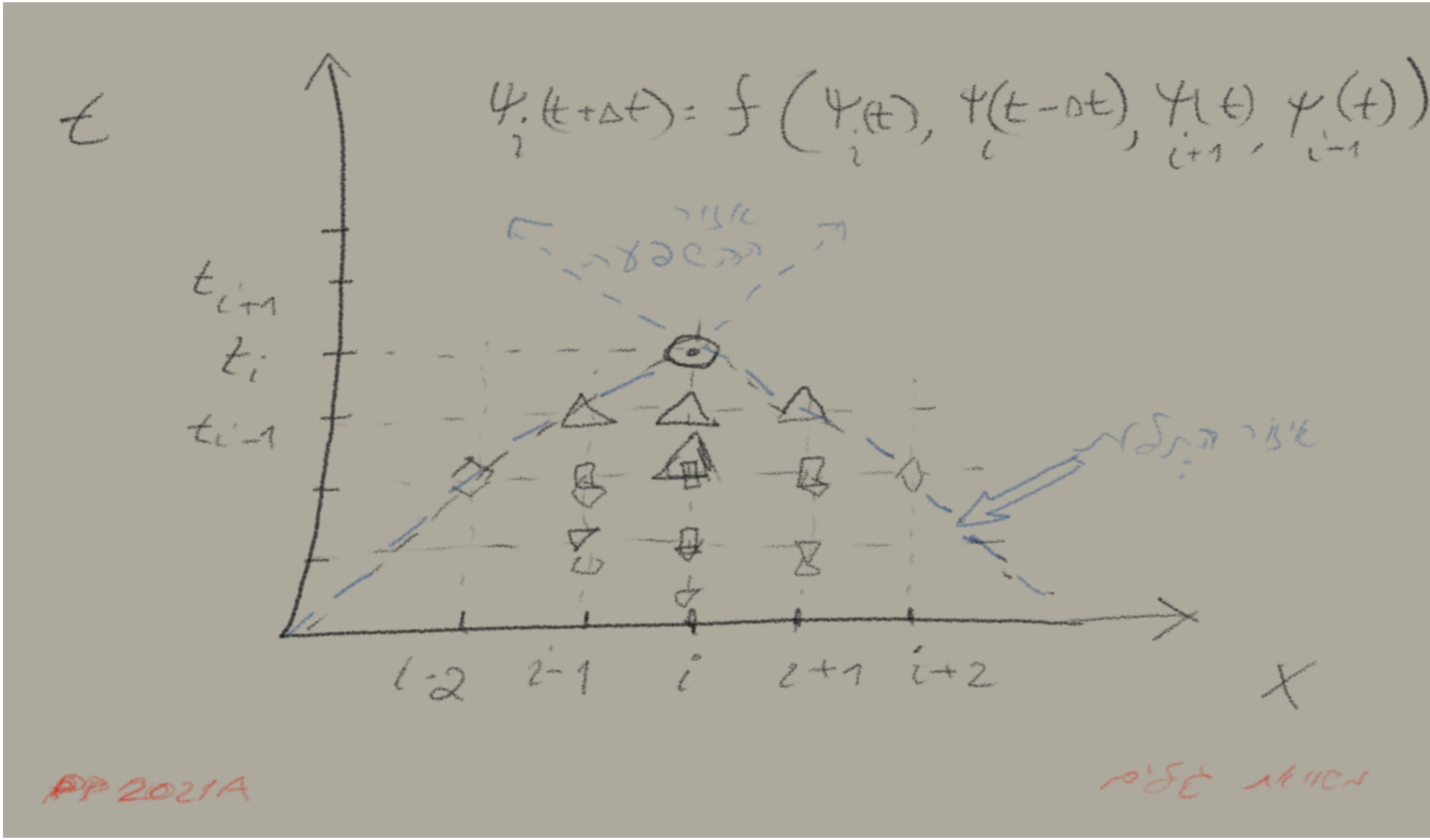
$$\frac{\psi_i(t-\Delta t)-2\psi_i(t)+\psi_i(t+\Delta t)}{c^2\Delta t^2}-\frac{\psi_{i-1}(t)-2\psi_i(t)+\psi_{i+1}(t)}{\Delta x^2}=0 \qquad (5\text{-}2)$$

Neglecting issues of solution stability, the straightforward scheme for solving this
em involves stepping sequentially from one time step to the next using the relation:

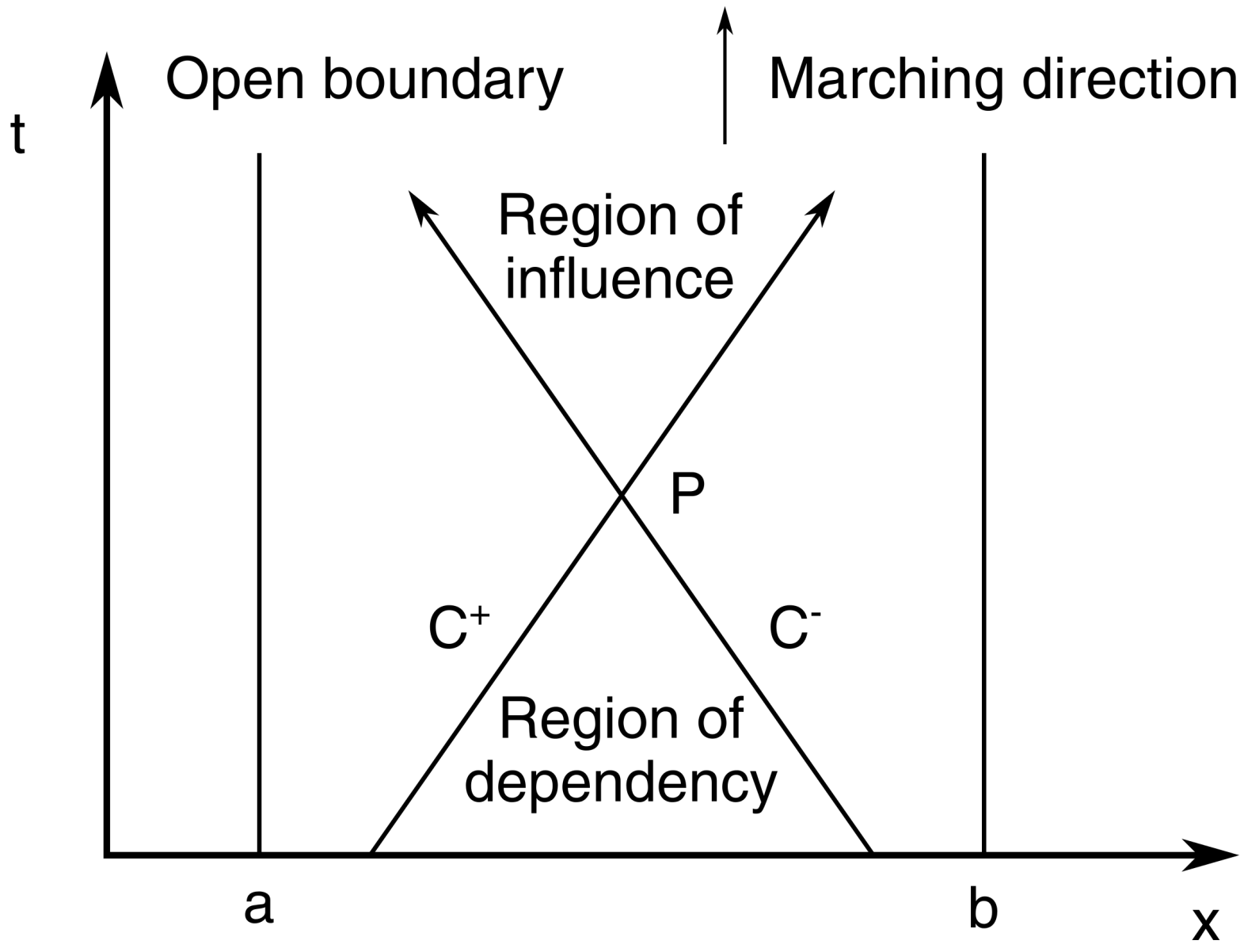
$$\psi_i(t+\Delta t)=2\psi_i(t)-\psi_i(t-\Delta t)+\tau^2[\psi_{i-1}(t)-2\psi_i(t)+\psi_{i+1}(t)] \qquad (5\text{-}3)$$

re $\tau=c\,\Delta t/\Delta x$.

The question now arises how best to divide the problem domain for effective con-
ent solution. With such a simple example, the answer is obvious because of the close
ogy between our problem and the Hadrian's Wall example. If we were to propose
ng each processor some subset of the time steps to calculate, this would correspond to
gning bricklayers to work on successive horizontal layers of the wall. Such a tem-



MyPaint/wave


Open boundary
Marching direction
t
Region of
influence
P
C+
C-
Region of
dependency
a
b
x

CFL CONDITION

When we discretize the time, the step must be less than the time it takes for the information to propagate across a single zone.

This is called **CFL**(Courant–Friedrichs–Lewy) condition

$$\Delta t \leq \frac{\Delta x}{v}$$

$$C = \frac{v\Delta t}{\Delta x}$$

$$C \leq 1$$

Necessary condition for stability

Über die partiellen Differenzengleichungen der mathematischen Physik (*)
Courant, R.; Lewy, H.; Friedrichs, K.
in: Mathematische Annalen | Mathematische, 1928

(*) About the partial difference equations of the mathematical physics

Über die partiellen Differenzengleichungen der mathematischen Physik.

Von

R. Courant, K. Friedrichs und H. Lewy in Göttingen.

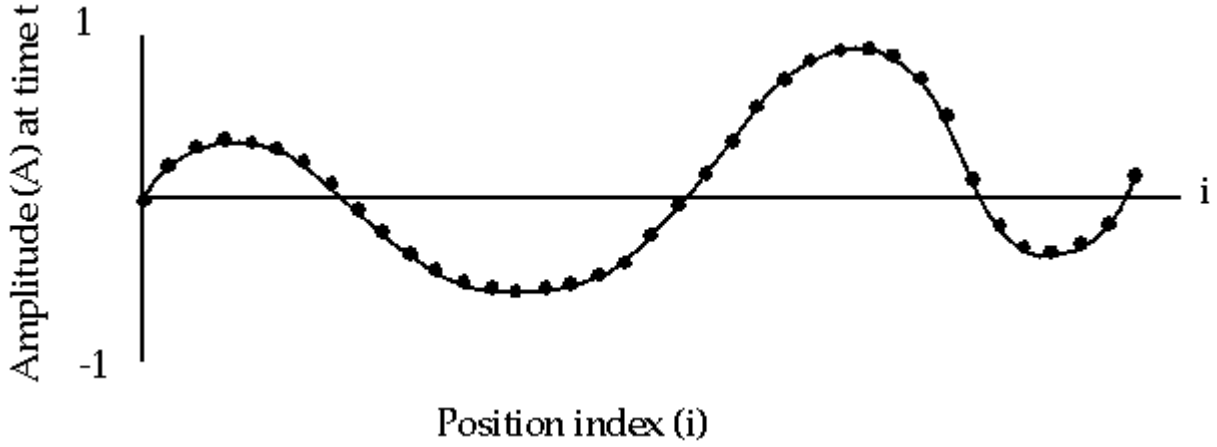
Ersetzt man bei den klassischen linearen Differentialgleichungsproblemen der mathematischen Physik die Differentialquotienten durch Differenzenquotienten in einem — etwa rechtwinklig angenommenen — Gitter, so gelangt man zu algebraischen Problemen von sehr durchsichtiger Struktur. Die vorliegende Arbeit untersucht nach einer elementaren Diskussion dieser algebraischen Probleme vor allem die Frage, wie sich die Lösungen verhalten, wenn man die Maschen des Gitters gegen Null streben läßt. Dabei beschränken wir uns vielfach auf die einfachsten, aber typischen Fälle, die wir derart behandeln, daß die Anwendbarkeit der Methoden auf allgemeinere Differenzengleichungen und solche mit beliebig vielen unabhängigen Veränderlichen deutlich wird.

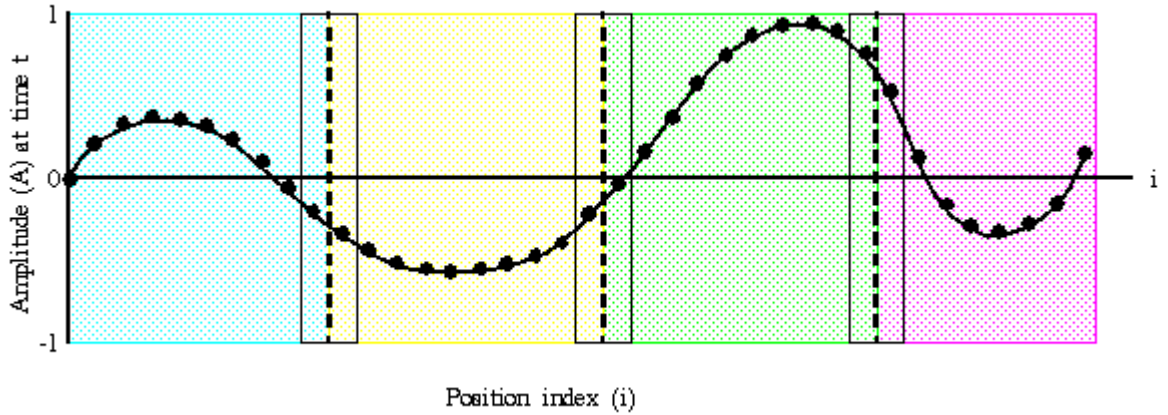
Entsprechend den für Differentialgleichungen geläufigen Fragestellungen behandeln wir Randwert- und Eigenwertprobleme für elliptische Differenzengleichungen und das Anfangswertproblem für hyperbolische bzw. parabolische Differenzengleichungen. Wir werden an einigen typischen Beispielen beweisen, daß der Grenzübergang stets möglich ist, nämlich daß die Lösungen der Differenzengleichungen gegen die Lösungen der entsprechenden Differentialgleichungsprobleme konvergieren; ja wir werden sogar erkennen, daß bei elliptischen Gleichungen i. a. die Differenzenquotienten beliebig hoher Ordnung gegen die entsprechenden Differentialquotienten streben. Die Lösbarkeit der Differentialgleichungsprobleme setzen wir nirgends voraus; vielmehr erhalten wir durch den Grenzübergang hierfür einen einfachen Beweis [1]). Während aber beim elliptischen

---

[1]) Unsere Beweismethode läßt sich ohne Schwierigkeit so erweitern, daß sie bei beliebigen linearen elliptischen Differentialgleichungen das Rand- und Eigenwertproblem und bei beliebigen linearen hyperbolischen Differentialgleichungen das Anfangswertproblem zu lösen gestattet.

See also:
https://www.dsc.soic.indiana.edu/sites/default/files/jsuwaveequation05.ppt





https://computing.llnl.gov/tutorials/parallel_comp/#ExamplesWave

find out number of tasks and task identities

#Identify left and right neighbors
```
left_neighbor  = mytaskid - 1
right_neighbor = mytaskid + 1
if mytaskid = first then left_neigbor = last
if mytaskid = last then right_neighbor = first
```

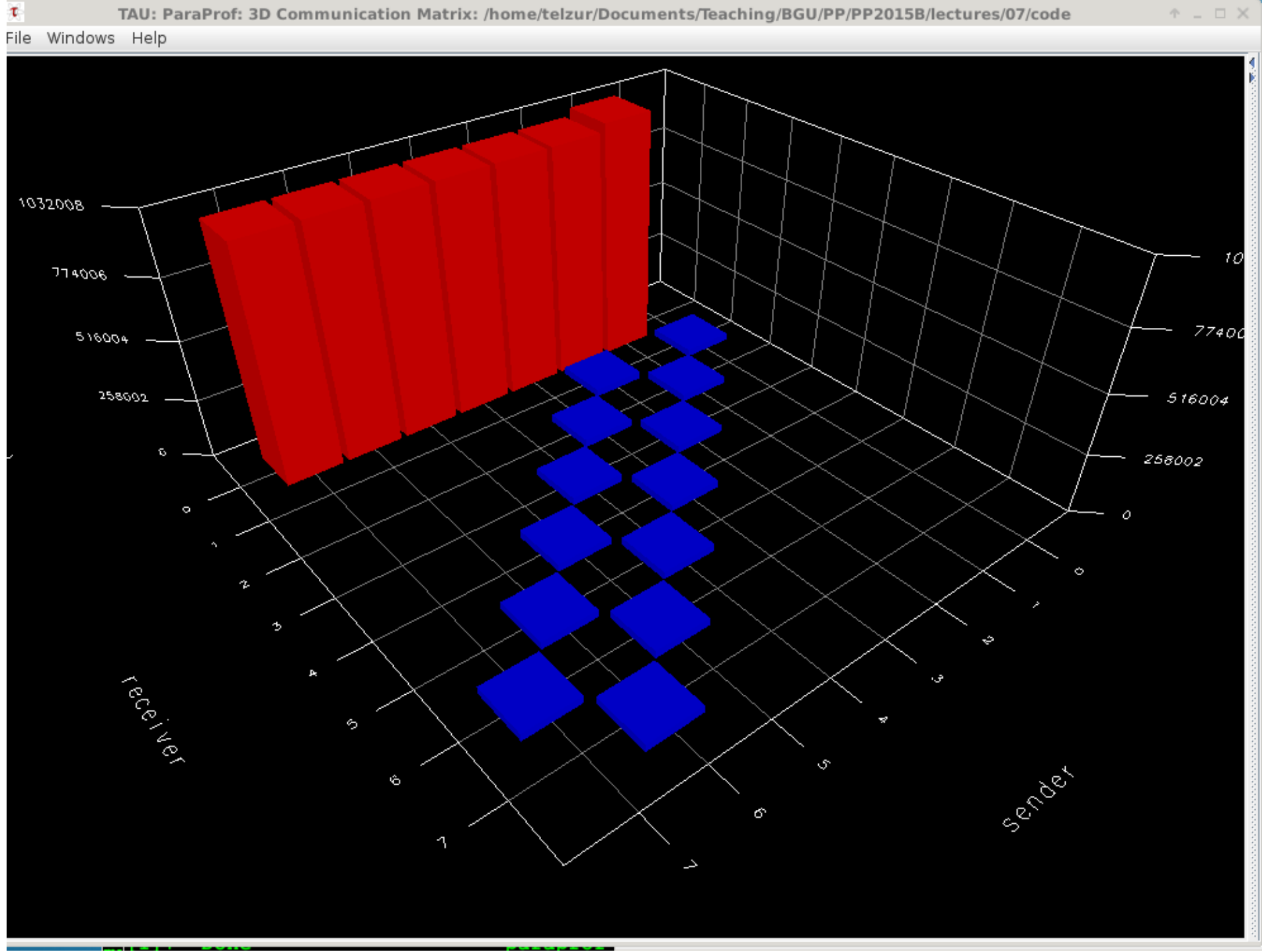
find out if I am MASTER or WORKER
if I am MASTER
  initialize array
  send each WORKER starting info and subarray
else if I am WORKER`
  receive starting info and sub-array from MASTER
endif

#Update values for each point along string
#In this example the master participates in calculations
do t = 1, nsteps
  send left endpoint to left neighbor
  receive left endpoint from right neighbor
  send right endpoint to right neighbor
  receive right endpoint from left neighbor

```
#Update points along line
  do i = 1, npoints
    newval(i) = (2.0 * values(i)) - oldval(i)
    + (sqtau * (values(i-1) - (2.0 * values(i)) +
values(i+1)))
  end do

 do i = 1, npoints
    values(i)=newval(i)
 end do

end do

#Collect results and write to file
if I am MASTER
  receive results from each WORKER
  write results to file
else if I am WORKER
  send results to MASTER
endif
```

Profiling with TAU

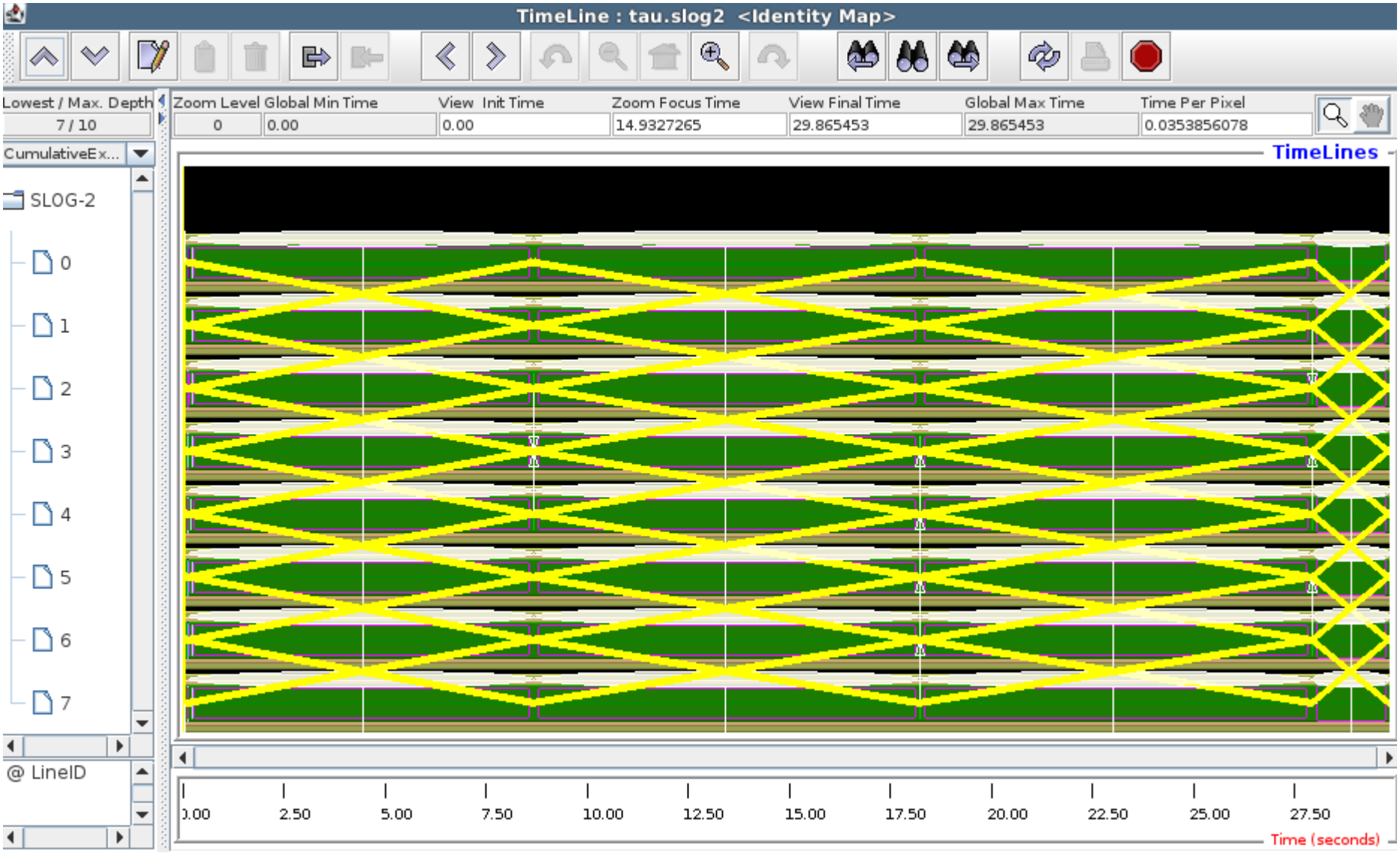
```
#> export TAU_COMM_MATRIX=1


#> export
TAU_MAKEFILE=/media/telzur/.../Download/tau-2.24/x86_64/lib/
Makefile.tau-callpath-mpi-python-pdt-openmp-opari


 #> tau_cc.sh -g -o wave ./wave.c

#> mpirun -np 8 ./wave
Wave solution running with 8 processes

0: points = 1000000, running for 30 seconds
points / second: 143.0M (17.9M per process)
compute / communicate efficiency: 90% | 94% | 99%

Points for validation:
0:0.00  200000:0.95  400000:0.59  600000:-0.59  800000:-0.95  999999:0.00
wave finished
```


TAU: ParaProf: 3D Communication Matrix: /home/telzur/Documents/Teaching/BGU/PP/PP2015B/lectures/07/code
File
Windows
Help
1032008
774006
516004
258002
0
receiver
sender
0
1
2
3
4
5
6
7

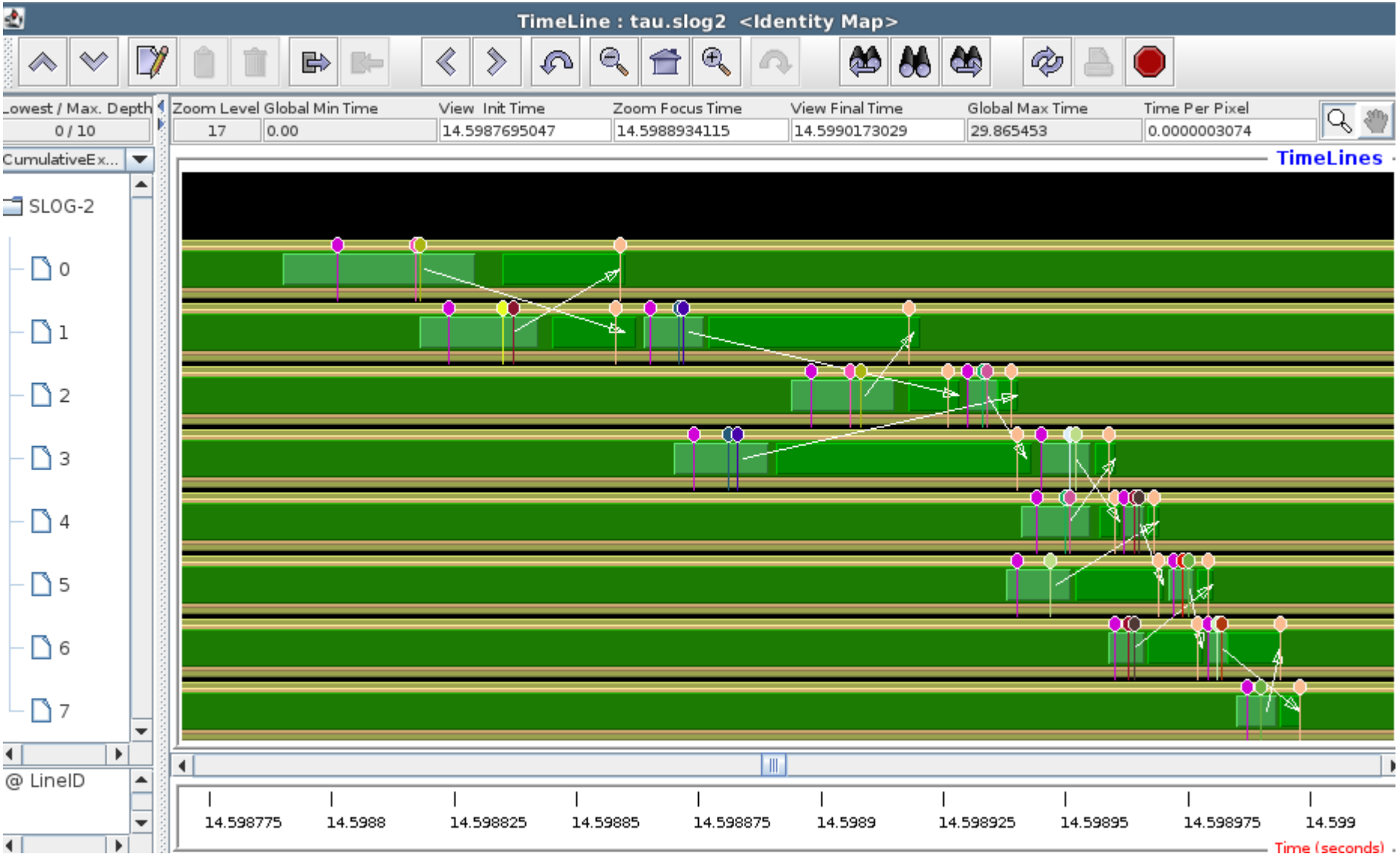
שימוש ב-tau ליצירת קובץ jumpshot

עבור המכונה הוירטואלית:

```
export TAU_TRACE=1
tau_cc.sh -g -o wave ./wave.c
mpirun -np 8 ./wave
tau_treemerge
tau2slog2 tau.trc tau.edf -o tau.slog2
jumpshot ./tau.slog2 &
```

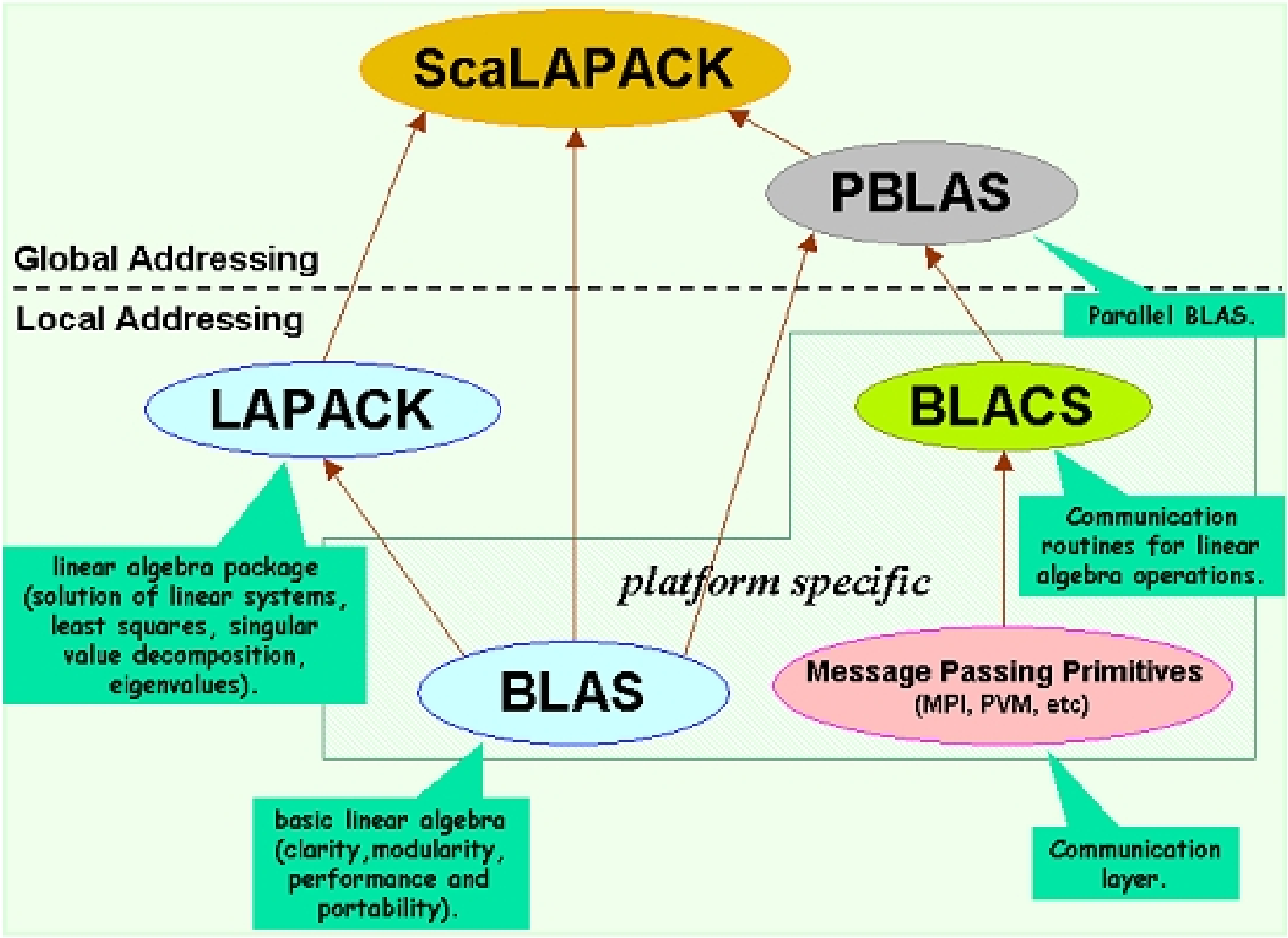

TimeLine : tau.slog2 <Identity Map>
Lowest / Max. Depth
7 / 10
Zoom Level
0
Global Min Time
0.00
View Init Time
0.00
Zoom Focus Time
14.9327265
View Final Time
29.865453
Global Max Time
29.865453
Time Per Pixel
0.0353856078
CumulativeEx...
SLOG-2
0
1
2
3
4
5
6
7
@ LineID
TimeLines
0.00
2.50
5.00
7.50
10.00
12.50
15.00
17.50
20.00
22.50
25.00
27.50
Time (seconds)

Zoom in


TimeLine : tau.slog2 <Identity Map>
Lowest / Max. Depth
0 / 10
Zoom Level
17
Global Min Time
0.00
View Init Time
14.5987695047
Zoom Focus Time
14.5988934115
View Final Time
14.5990173029
Global Max Time
29.865453
Time Per Pixel
0.0000003074
CumulativeEx...
SLOG-2
0
1
2
3
4
5
6
7
@ LineID
TimeLines
14.598775
14.5988
14.598825
14.59885
14.598875
14.5989
14.598925
14.59895
14.598975
14.599
Time (seconds)

Linear Algebra Libraries

Scalapack


ScaLAPACK
PBLAS
Global Addressing
Local Addressing
Parallel BLAS.
LAPACK
BLACS
linear algebra package (solution of linear systems, least squares, singular value decomposition, eigenvalues).
platform specific
Communication routines for linear algebra operations.
BLAS
Message Passing Primitives
(MPI, PVM, etc)
basic linear algebra (clarity, modularity, performance and portability).
Communication layer.

BLAS

Level 1: vector-vector operations

ex. $\vec{\mathbf{y}} \leftarrow \alpha\vec{\mathbf{x}} + \vec{\mathbf{y}}$: ?axpy,?=s,d,c,z

Level 2: matrix-vector operations

ex. $\vec{\mathbf{y}} \leftarrow \alpha\mathbf{A}\vec{\mathbf{x}} + \beta\vec{\mathbf{y}}$: ?gemv

Level 3: matrix-matrix operations

ex. $\mathbf{C} \leftarrow \mathbf{AB} + \alpha\mathbf{C}$: ?gemm,?symm

- Standard but bit cryptic (saxpy,dgemm,...) and 'very fortran'

www.chpc.utah.edu/attachments/20110113.00/MatLib06.pdf

BLAS

Center for High-Performance Computing @ the University of Utah

- Basic Linear Algebra Subroutines
- Level 1 – vector-vector operations
  - CAXPY(N,A,X,XMAX,Y,YMAX)
- Level 2 – matrix-vector operations
  - ZGEMV('N',M,1.,A,MMAX,X,XMAX,0.,Y,YMAX)
- Level 3 – matrix-matrix operations
  - DGEMM('N','N',M,R,N,1.,A,MMAX,B,MMAX,0.,C,MMAX)
- More information
- http://netlib2.cs.utk.edu/blas/

How to decode BLAS names

Center for High-Performance Computing @ the University of Utah

- First letter – precision (S,D,C,Z)
- Level 1 – next letters – operation

  e.g. `AXPY` - constant times vector plus vector
- Level 2, 3
- 2nd and 3rd letters – type of vector

  e.g. `GE` - general matrix
- following letters – operation

  e.g. `MV` – matrix-vector product

BLAS example


```
PARAMETER ( MMAX = 10, NMAX = 10, KMAX = 10)
REAL A(MMAX,NMAX)
REAL B(NMAX,KMAX)
REAL C(MMAX,KMAX)
REAL X(MMAX), Y(NMAX)
INTEGER M,N,K

M = 2; N = 4; K = 3
CALL SGEMM('N','N',M,N,K,1.,A,MMAX,B,NMAX,0.,C,MMAX)
CALL SGEMV('N',M,N,1.,A,MMAX,X,MMAX,0.,Y,NMAX)
```

$$\mathbf{C} = \mathbf{AB}$$
$$\bar{y} = \mathbf{A}\bar{x}^{\mathrm{T}}$$

In C, must use CBLAS – C bindings for BLAS
For gcc, also must link fortran to C transform library:

```
gcc matmul_1.c -L/home/mcuma/linal/CBLAS/lib/LINUX -lcblas -
lblas -L/usr/lib/gcc-lib/i386-redhat-linux/2.96 -lg2c -o
matmul_1
```

OpenBLAS
An optimized BLAS library based on GotoBLAS

http://www.openblas.net/

OpenBLAS → BLIS הפצה יותר חדשה

https://github.com/flame/blis

LAPACK

Center for High-Performance Computing @ the University of Utah

- **systems of linear equations, eigenvalue problems, …**

http://www.netlib.org/lapack/

- **ATLAS also contains some LAPACK routines**
- **ACML contains full LAPACK**

  use routines from ATLAS if possible, or, for ease of use, just use ACML

How to decode LAPACK names

Center for High-Performance Computing
@ the University of Utah

- **XYYZZZ name format**
- **X – precision (S,D,C,Z)**
- **YY –type of matrix**

  e.g. `ge` **– general,** `he` **– Hermitian, …**
- **ZZZ – computation to be performed**

  e.g. `trf` **– LU factorization with partial pivoting**

  `tri` **– inverse of matrix**

  `sv` **– system of linear equations AX=B**

  `ev` **– eigenvalues and eigenvectors**

ScaLAPACK

Center for High-Performance Computing @ the University of Utah

- parallel distributed version of LAPACK
- MPI_BLACS for communication
- PBLAS for BLAS

http://www.netlib.org/scalapack/

- linking must include paths to all libraries – better to use Makefile based on the testing routines from the distribution

In the next slides:

Examples:

#	Program name	Location on my laptop	Location in the hobbits	Goal
1	dgemm_example.c	.../07/code	~/mpi	cblas_dgemm
2	~~dgemm_demo.c~~	~~.../07/code~~		~~cblas_dgemm~~
3	dgemm-mpi.c	.../07/code		cblas+mpi mm
4	C and FORTRAN mixing	~/calling_fortran_from_c		
5	pdgmv.c	OK		
6	mm_pblas.c	OK		

Program #1: dgemm_example.c

dgemm_example.c **This is a Serial code**

File is under `~/.../lecture07/code/dgemm_example.c`
and in the hobbit under: `~/mpi/dgemm_example.c`
With openblas:

```
gcc -o dgemm_example ./dgemm_example.c -lopenblas
```

This program does not work out of the box on the BGU-VM
<Teacher note: I installed *openblas* on my BGU-VM and it works>

Compilation on the hobbits:

```
gcc -o dgemm_example ./dgemm_example.c \
        -L/usr/lib64/atlas -lcblas
```

Example based on:
http://software.intel.com/sites/products/documentation/doclib/mkl_sa/11/tutorials/mkl_mmx_c/tutorial_mkl_mmx_c.pdf

The arguments of DGEMM

```
cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,
            m, n, k, alpha, A, k, B, n, beta, C, n);
```

The arguments provide options for how Intel MKL performs the operation. In this case:

CblasRowMajor
Indicates that the matrices are stored in row major order, with the elements of each row of the matrix stored contiguously as shown in the figure above.

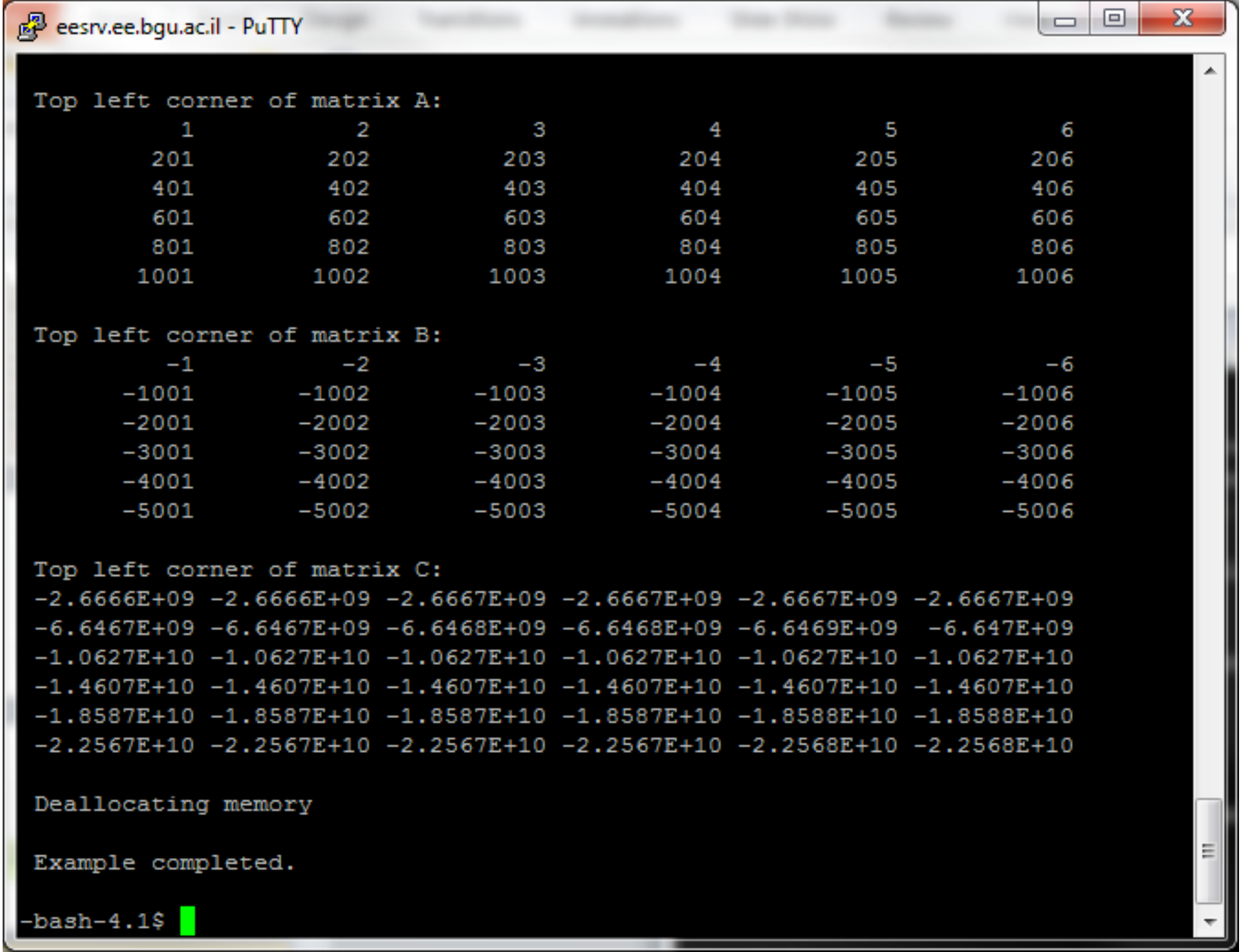
CblasNoTrans
Enumeration type indicating that the matrices A and B should not be transposed or conjugate transposed before multiplication.

m, n, k - Integers indicating the size of the matrices:
A: m rows by k columns
B: k rows by n columns
C: m rows by n columns

alpha, beta - Real value used to scale the product of matrices A and B.

eesrv.ee.bgu.ac.il - PuTTY

```
-bash-4.1$ ./dgemm_example

 This example computes real matrix C=alpha*A*B+beta*C using
 cblass dgemm, where A, B, and  C are matrices and
 alpha and beta are double precision scalars

 Initializing data for matrix multiplication C=A*B for matrix
 A(2000x200) and matrix B(200x1000)

 Allocating memory for matrices aligned on 64-byte boundary for better
 performance

 Intializing matrix data

 Computing matrix product using dgemm function via CBLAS interface


 Computations completed.

 Top left corner of matrix A:
            1            2            3            4            5            6
          201          202          203          204          205          206
          401          402          403          404          405          406
          601          602          603          604          605          606
```


eesrv.ee.bgu.ac.il - PuTTY
Top left corner of matrix A:
1 2 3 4 5 6
201 202 203 204 205 206
401 402 403 404 405 406
601 602 603 604 605 606
801 802 803 804 805 806
1001 1002 1003 1004 1005 1006
Top left corner of matrix B:
-1 -2 -3 -4 -5 -6
-1001 -1002 -1003 -1004 -1005 -1006
-2001 -2002 -2003 -2004 -2005 -2006
-3001 -3002 -3003 -3004 -3005 -3006
-4001 -4002 -4003 -4004 -4005 -4006
-5001 -5002 -5003 -5004 -5005 -5006
Top left corner of matrix C:
-2.6666E+09 -2.6666E+09 -2.6667E+09 -2.6667E+09 -2.6667E+09 -2.6667E+09
-6.6467E+09 -6.6467E+09 -6.6468E+09 -6.6468E+09 -6.6469E+09 -6.647E+09
-1.0627E+10 -1.0627E+10 -1.0627E+10 -1.0627E+10 -1.0627E+10 -1.0627E+10
-1.4607E+10 -1.4607E+10 -1.4607E+10 -1.4607E+10 -1.4607E+10 -1.4607E+10
-1.8587E+10 -1.8587E+10 -1.8587E+10 -1.8587E+10 -1.8588E+10 -1.8588E+10
-2.2567E+10 -2.2567E+10 -2.2567E+10 -2.2567E+10 -2.2568E+10 -2.2568E+10
Deallocating memory
Example completed.
-bash-4.1$

ScaLAPACK

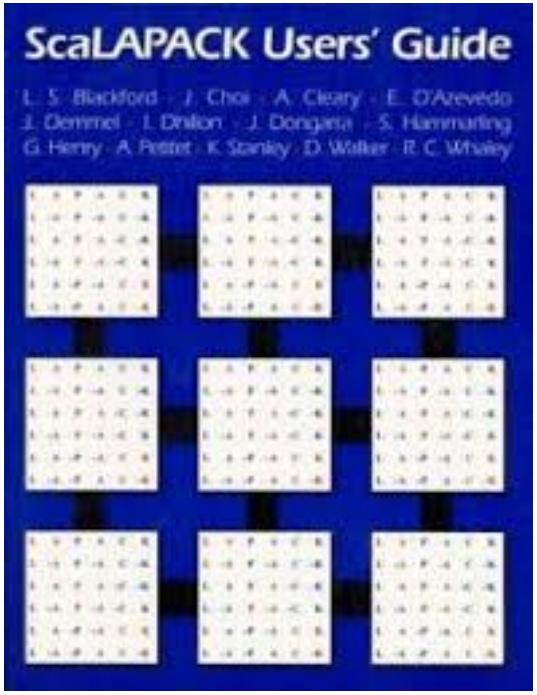
Scalable Linear Algebra PACKage

http://www.netlib.org/scalapack/

Hands-On Exercises for ScaLAPACK

http://acts.nersc.gov/scalapack/hands-on/

L A P A C K
L -A P -A C -K
L A P A -C -K
L -A P -A -C K
L A -P -A C K
L -A -P A C -K



ScaLAPACK is 2.1.0.0, released in November 16, 2019

ScaLAPACK

- **ScaLAPACK** is for solving **dense linear systems** and computing eigenvalues for dense matrices

Description

- A Scalable Linear Algebra Package (ScaLAPACK)
- A library of linear algebra routines for MPP (Massively Parallel Processing) and NOW (Network of Workstations) machines
- Language : FORTRAN
- Dense Matrix Problem Solvers
  - Linear Equations
  - Least Squares
  - Eigenvalue

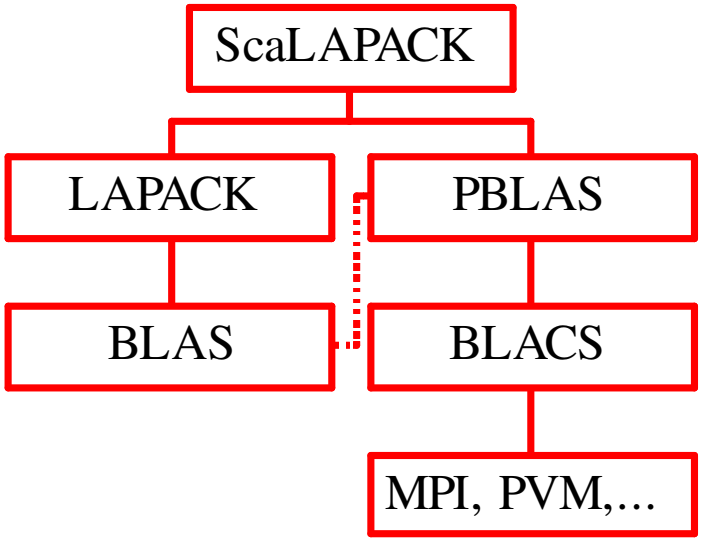
Technical Information

- ScaLAPACK web page
  - http://www.netlib.org/scalapack
- ScaLAPACK User's Guide
  - http://netlib.org/scalapack/slug/scalapack_slug.html
- Technical Working Notes
  - http://www.netlib.org/lapack/lawn

Packages

- **LAPACK**
  - Routines are serial Linear Algebra solvers, utilities, etc.
- **BLAS**
  - Serial work-horse routines (vector, vector-matrix & matrix-matrix)
- **PBLAS**
  - Parallel counterparts of BLAS. relies on BLACS for blocking and moving data.

Dependencies & Parallelism of Component Packages



Serial Parallel

Packages (cont.)

Basic Linear Algebra Communication Subprograms

- BLACS
  - Constructs 1-D & 2-D processor grids for matrix decomposition and nearest neighbor communication patterns. Uses message passing libraries (e.g. MPI) for data movement.

2D-Grid, Scope, Context
• Processes are arranged in a logical 2D-grid. • Each process is a member of three scopes:
– ’All’: All processes in the grid
– ’Row’: All processes on the same row of the grid
– ’Column’: All processes on the same column of the grid
• BLACS communication is tied to a **context** (think of MPI communicators) which is an integer.

BLACS – Setup (FORTRAN)

- Initializing BLACS:
  - **`CALL BLACS_PINFO(ME, NP)`**

- Initializing context: – CALL BLACS_GET(0, 0, CTXT)
  - **`CALL BLACS_GRIDINIT(CTXT, 'Row', P, Q)`**
  - **`CALL BLACS_GRIDINFO(CTXT, P, Q, MYROW, MYCOL)`**

- Getting someones rank from coordinates
  - **`RANK = BLACS_PNUM(CTXT, ROW, COL)`**

- Getting someones coordinates from rank
  - **`CALL BLACS_PCOORD(CTXT, RANK, ROW, COL)`**

- Exiting BLACS
  - **`CALL BLACS_EXIT(0)`**

Data Partitioning

- Initialize BLACS routines
  - Set up 2-D mesh (mp, np)
  - Set (CorR) Column or Row Major Order
  - Call returns context handle (icon)
  - Get row & column through gridinfo (myrow,mycol)

- **`blacs_gridinit( icon, CorR, mp, np )`**
- **`blacs_gridinfo( icon, mp, np, myrow, mycol)`**

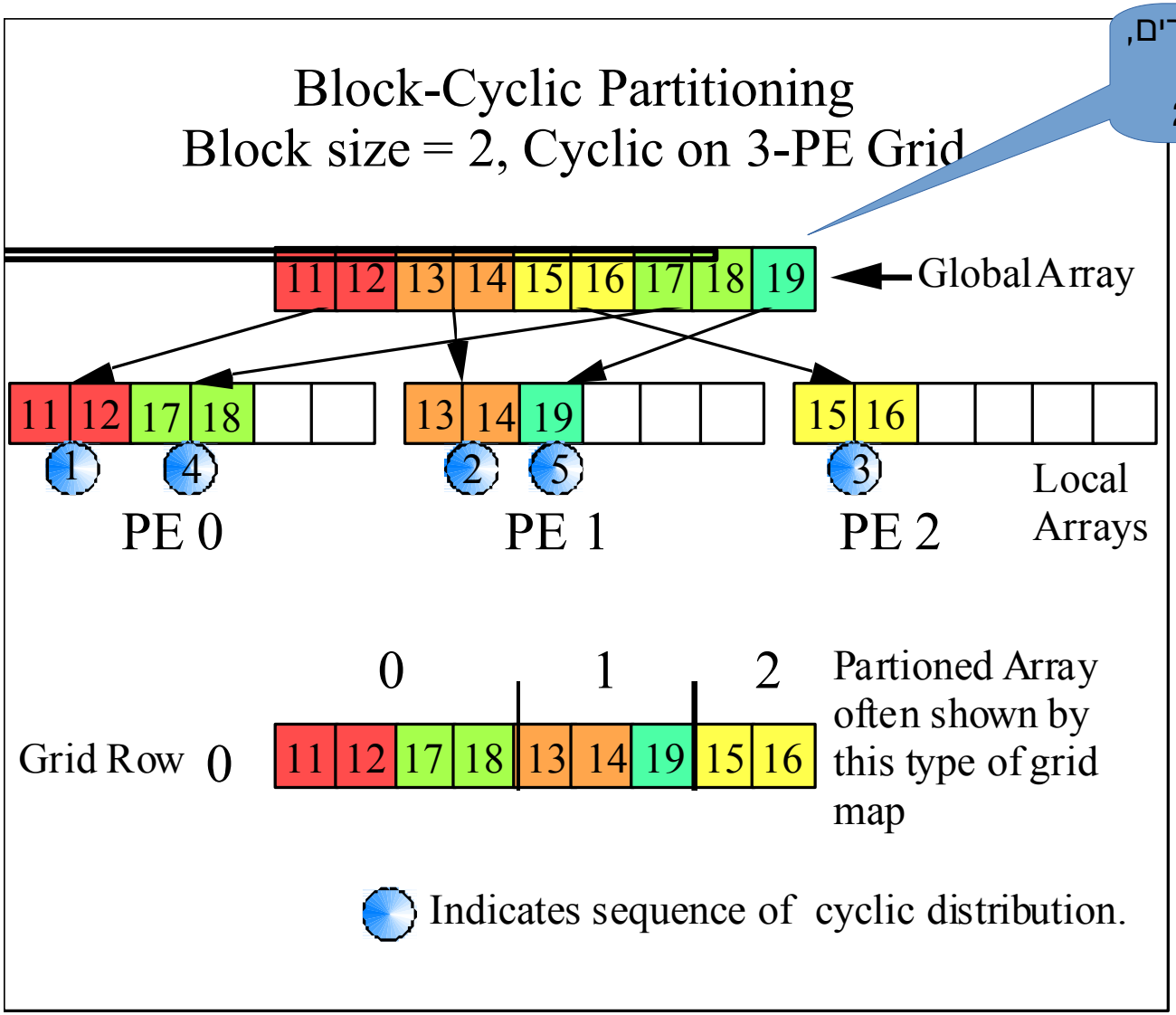
Data Partitioning (cont.)

- Block-cyclic Array Distribution.
  - Block: (groups of sequential elements)
  - Cyclic: (round-robin assignment to PEs)

דוגמאות להקצאת המעבדים על המערכים/מטריצות:

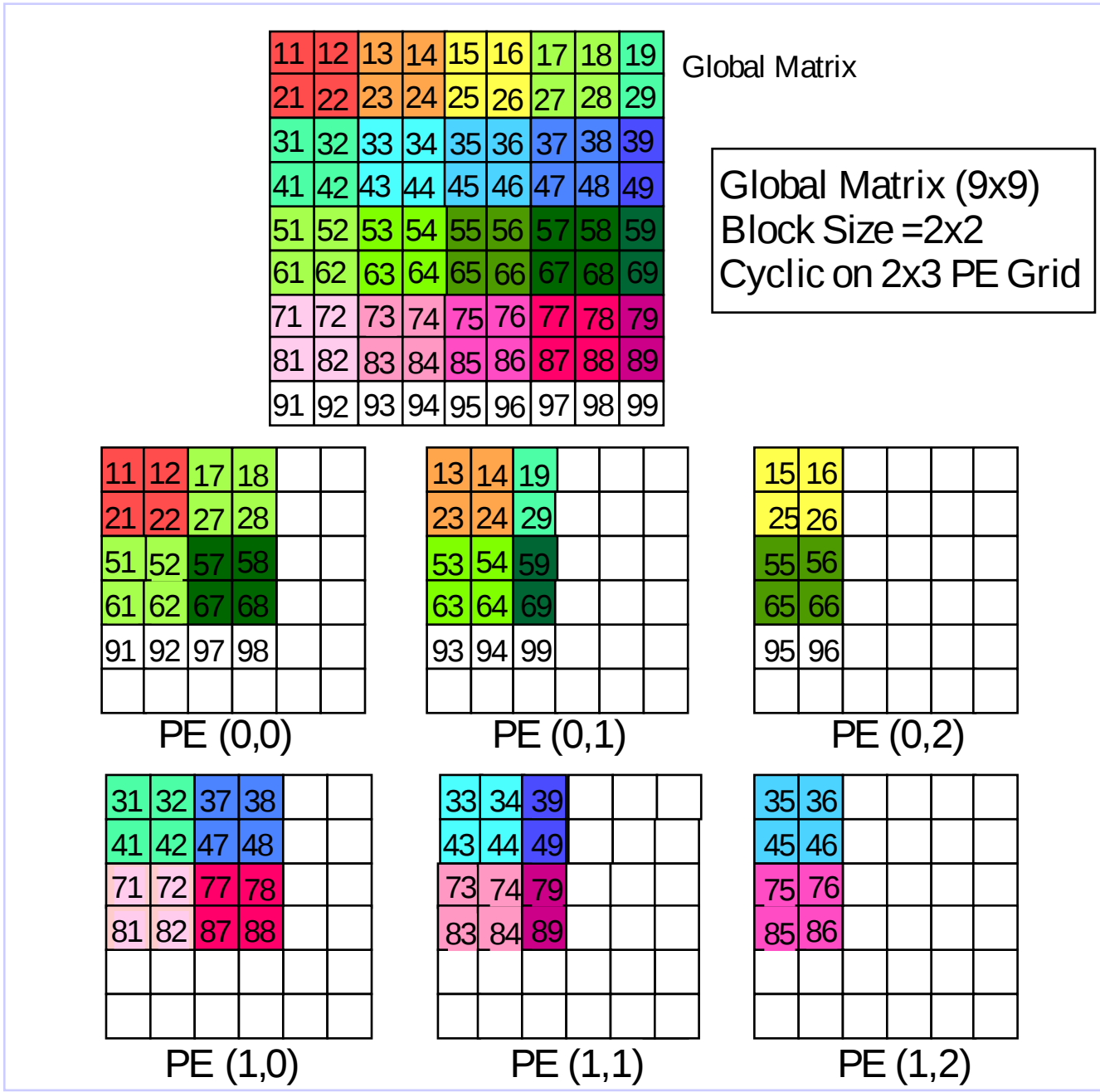
- Example 1-D Partitioning:
  - Global 9-element array distributed on a 3-PE grid with 2 element blocks: block(2)-cyclic(3)
  - PE = Processing Element

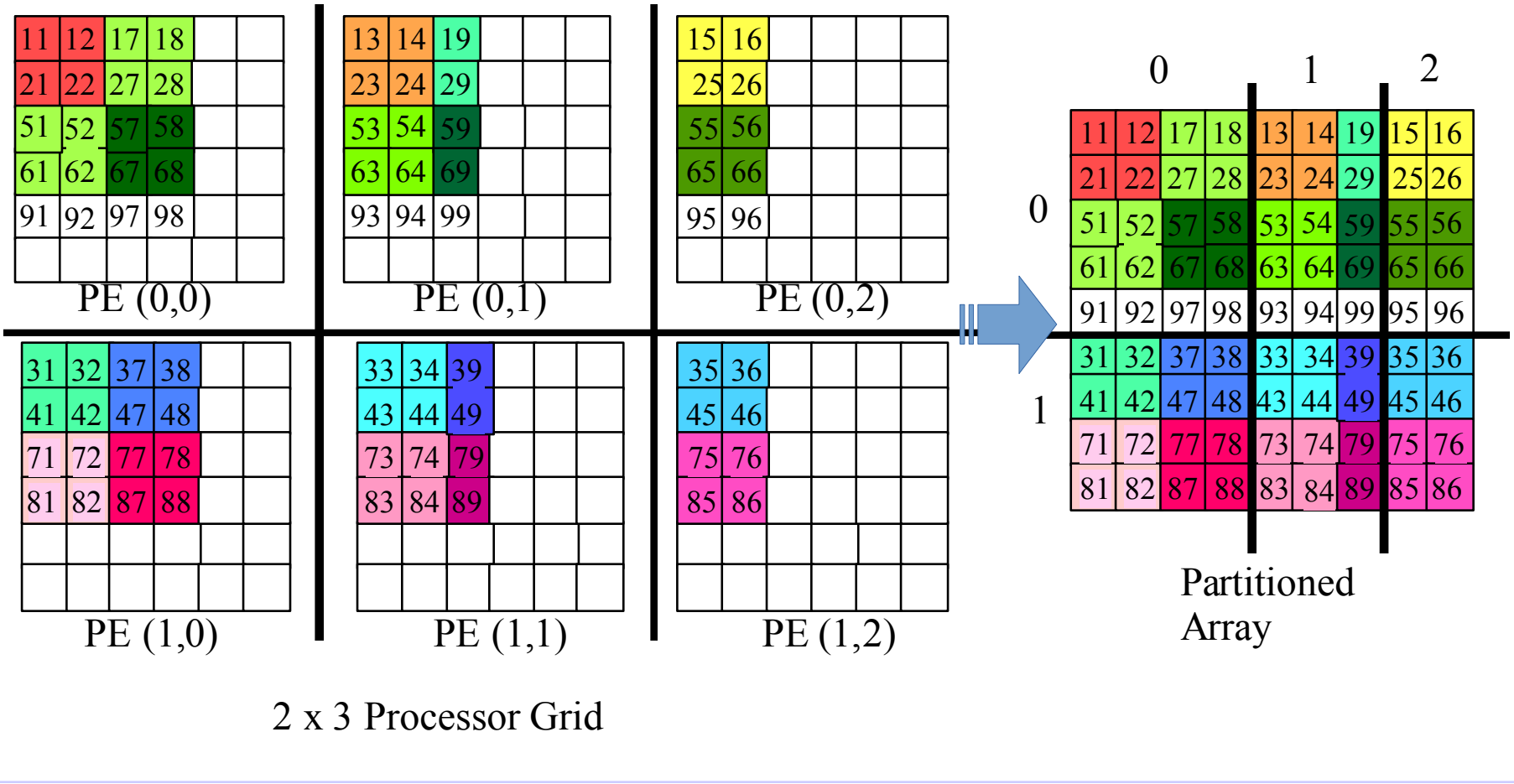
Block-Cyclic Partitioning



2-D Partitioning

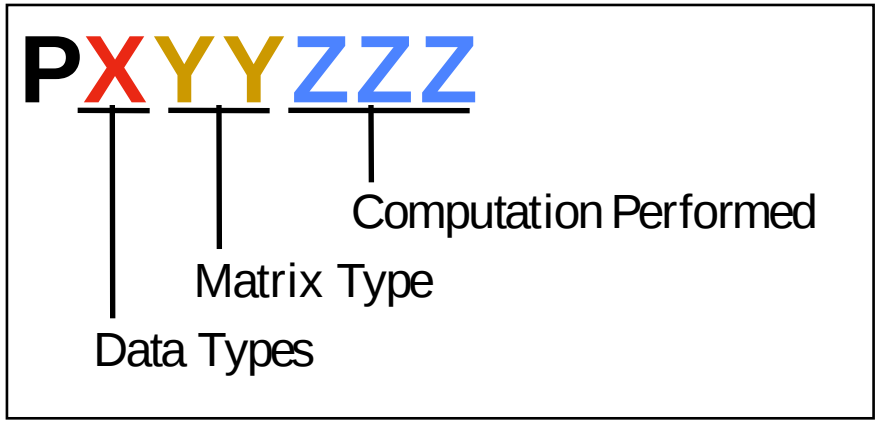
- Example 2-D Partitioning
  - A(9,9) mapped onto PE(2,3) grid block(2,2)
  - Use 1-D example to distribute column elements in each row, a block(2)-cyclic(3) distribution.
  - Assign rows to first PE dimension by block(2)-cyclic(2) distribution.


11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
41 42 43 44 45 46 47 48 49
51 52 53 54 55 56 57 58 59
61 62 63 64 65 66 67 68 69
71 72 73 74 75 76 77 78 79
81 82 83 84 85 86 87 88 89
91 92 93 94 95 96 97 98 99
Global Matrix
Global Matrix (9x9)
Block Size =2x2
Cyclic on 2x3 PE Grid
11 12 17 18
21 22 27 28
51 52 57 58
61 62 67 68
91 92 97 98
PE (0,0)
13 14 19
23 24 29
53 54 59
63 64 69
93 94 99
PE (0,1)
15 16
25 26
55 56
65 66
95 96
PE (0,2)
31 32 37 38
41 42 47 48
71 72 77 78
81 82 87 88
PE (1,0)
33 34 39
43 44 49
73 74 79
83 84 89
PE (1,1)
35 36
45 46
75 76
85 86
PE (1,2)


11 12 17 18
21 22 27 28
51 52 57 58
61 62 67 68
91 92 97 98
PE (0,0)
13 14 19
23 24 29
53 54 59
63 64 69
93 94 99
PE (0,1)
15 16
25 26
55 56
65 66
95 96
PE (0,2)
31 32 37 38
41 42 47 48
71 72 77 78
81 82 87 88
PE (1,0)
33 34 39
43 44 49
73 74 79
83 84 89
PE (1,1)
35 36
45 46
75 76
85 86
PE (1,2)
2 x 3 Processor Grid
0
1
2
0
1
Partitioned Array

API (cont..)

- LAPACK equivalent names with P.



Data Type	Real (single)	Double	Complex	Double complex
X	S	D	C	Z

Matrix Types (**YY**) P**X**YYZZZ

- **DB** General Band (diagonally dominant-like)
- **DT** general Tridiagonal (Diagonally dominant-like)
- **GB** General Band
- **GE** GEneral matrices (e.g., unsymmetric, rectangular, etc.)
- **GG** General matrices, Generalized problem
- **HE** Complex Hermitian
- **OR** Orthogonal Real
- **PB** Positive definite Banded (symmetric or Hermitian)
- **PO** Positive definite (symmetric or Hermitian)
- **PT** Positive definite Tridiagonal (symmetric or Hermitian)
- **ST** Symmetric Tridiagonal Real
- **SY** SYmmetric
- **TR** TRiangular (possibly quasi-triangular)
- **TZ** TrapeZoidal
- **UN** UNitary complex

Mixing C and FORTRAN

- Demo on my laptop (BGU-VM).
  …/07/code/calling_fortran_from_c/cprog.c and ffunction.f

- Important! Column-major order vs. Row-major order: C => Row, Fortran=>Column

Content of a library

```
PP> ar -t /usr/lib/avr/lib/libm.a
acos.o
addsf3.o
addsf3x.o
asin.o
atan2.o
atan.o
cbrt.o
ceil.o
cmpsf2.o
copysign.o
cosh.o
cos.o
divsf3.o
divsf3x.o
exp.o
fixsfdi.o
fixsfsi.o
fixunssfsi.o
floatdisf.o
floatsisf.o
floatundisf.o
fdim.o
...
```

ספריה היא אוסף של קבצי obj
ארוזים יחד

```
telzur ~/calling_fortran_from_c $ more ./ffunction.f
      subroutine ffunction(a,b)
      a=3.0
      b=4.0
      end
```

```
telzur ~/calling_fortran_from_c $ more ./cprog.c
#include <stdio.h>
int main(void) {
    float a=1.0, b=2.0;

    printf("Before running Fortran function:\n");
    printf("a=%f\n",a);
    printf("b=%f\n",b);
    ffunction_(&a,&b);
    printf("After running Fortran function:\n");
    printf("a=%f\n",a);
    printf("b=%f\n",b);
    return 0;
}
```

```
telzur ~/calling_fortran_from_c $ gfortran -c ffunction.f
telzur ~/calling_fortran_from_c $ gcc -c cprog.c
telzur ~/calling_fortran_from_c $ gcc -o cprog cprog.o
ffunction.o
telzur ~/calling_fortran_from_c $ ./cprog
Before running Fortran function:
a=1.000000
b=2.000000
After running Fortran function:
a=3.000000
b=4.000000
```

Program #5: pdgemv.c

Parallel Linear Algebra Libraries: Example PDGEMV()

Example: PDGEMV()

PDGEMV() computes a distributed matrix-vector product

$y = \alpha \cdot A \cdot x + \beta \cdot y$

לדוגמה:

A= [1 4 7 10 13 ; 3 6 9 12 15 ; 5 8 11 14 17 ; 7 10 13 16 19; 9 12 15 18 21]

and $x=[1 ; 1; 0; 0; 1]^T$, x is a column vector,

Call PDGEMV() routine to compute y=Ax.

The right result is $y=[18; 24; 30; 36; 42]^T$

(T=transpose)

Code sample in C:

The function call in C is like,

```
double alpha = 1.0; double beta = 0.0;
   pdgemv_("N",&M,&M,&alpha,A,&ONE,&ONE,descA,x,&ONE,&
ONE,descx,&ONE,&beta,y,&ONE,& amp;ONE,descy,&ONE);
```

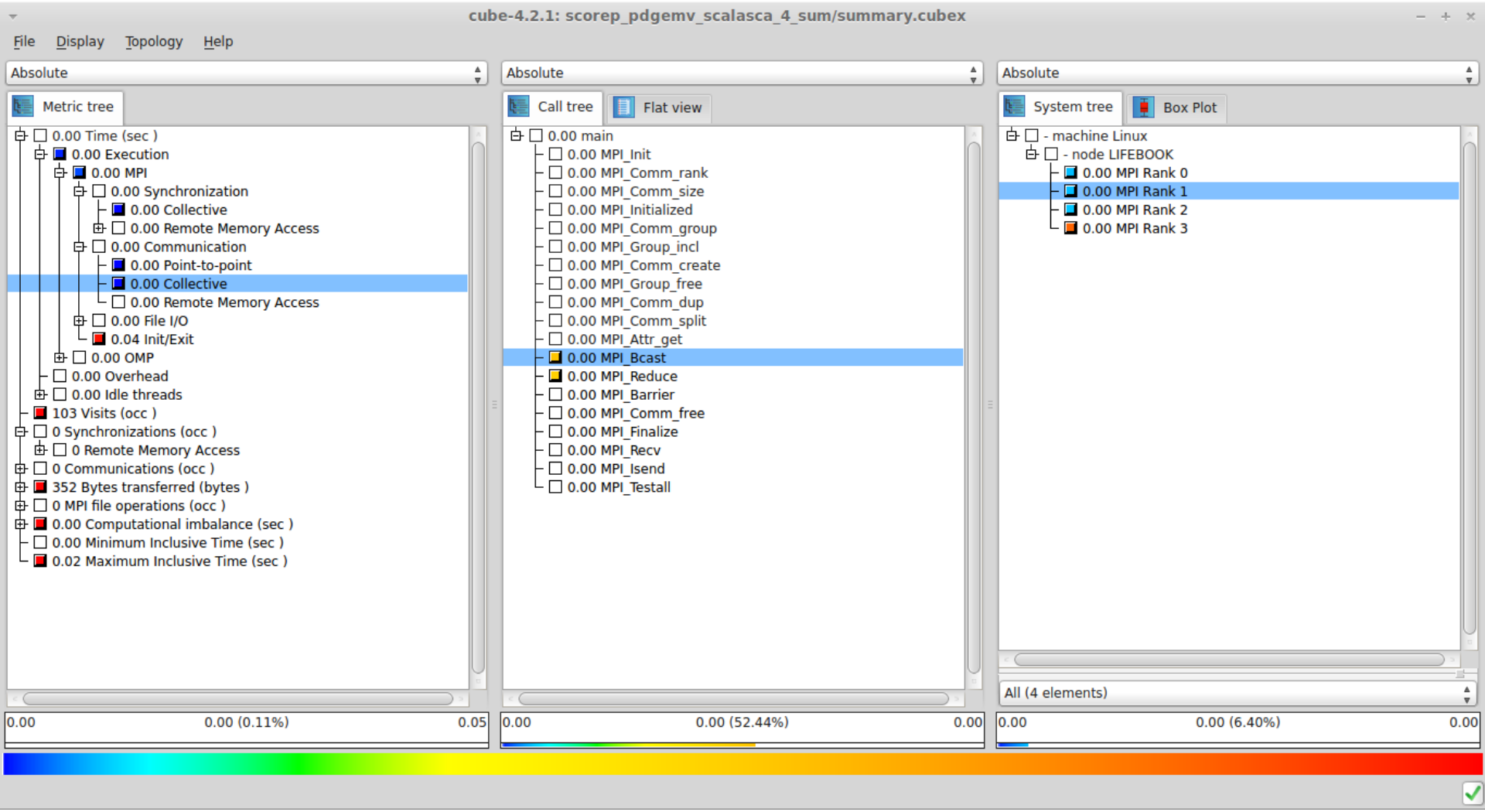
Purpose
=======
PDGEMV performs one of the matrix-vector operations
sub(Y) := alpha*sub(A) *sub(X) + beta*sub(Y), or
sub(Y) := alpha*sub(A)'*sub(X) + beta*sub(Y),
where
sub(A) denotes A(IA:IA+M-1,JA:JA+N-1).
When TRANS = 'N',
sub(X) denotes X(IX:IX,JX:JX+N-1), if INCX = M_X,
X(IX:IX+N-1,JX:JX), if INCX = 1 and INCX <> M_X,
and,
sub(Y) denotes Y(IY:IY,JY:JY+M-1), if INCY = M_Y,
Y(IY:IY+M-1,JY:JY), if INCY = 1 and INCY <> M_Y,
and, otherwise
sub(X) denotes X(IX:IX,JX:JX+M-1), if INCX = M_X,
X(IX:IX+M-1,JX:JX), if INCX = 1 and INCX <> M_X,
and,
sub(Y) denotes Y(IY:IY,JY:JY+N-1), if INCY = M_Y,
Y(IY:IY+N-1,JY:JY), if INCY = 1 and INCY <> M_Y.
Alpha and beta are scalars, and sub(X) and sub(Y) are subvectors
and sub(A) is an m by n submatrix.

pdgemv.c demo on the hobbits

```
-bash-4.1$ mpicc -o pdgemv ./pdgemv.c -L/usr/lib64/atlas -lcblas
-L/usr/local/plasma/lib -lscalapack -llapack
-bash-4.1$ mpirun -n 4 ./pdgemv
rank=0 18.00
rank=0 24.00
rank=0 42.00
rank=1 0.00
rank=1 0.00
rank=1 0.00
rank=2 30.00
rank=2 36.00
rank=3 0.00
rank=3 0.00
-bash-4.1$
```

This program cannot be executed out of the box on the BGU-VM version 10.20 because Scalapack isn't installed.
30/11/20 update: I can compile it on "My" BGU-VM after installing all the needed dependencies -see in a few slides. Make a demo.

pdgemv profiling



Program #6: mm_pblas.c

Demo of PDGEMM

- Parallel **DGEMM**

i.e. Parallel Matrix-Matrix Multiply

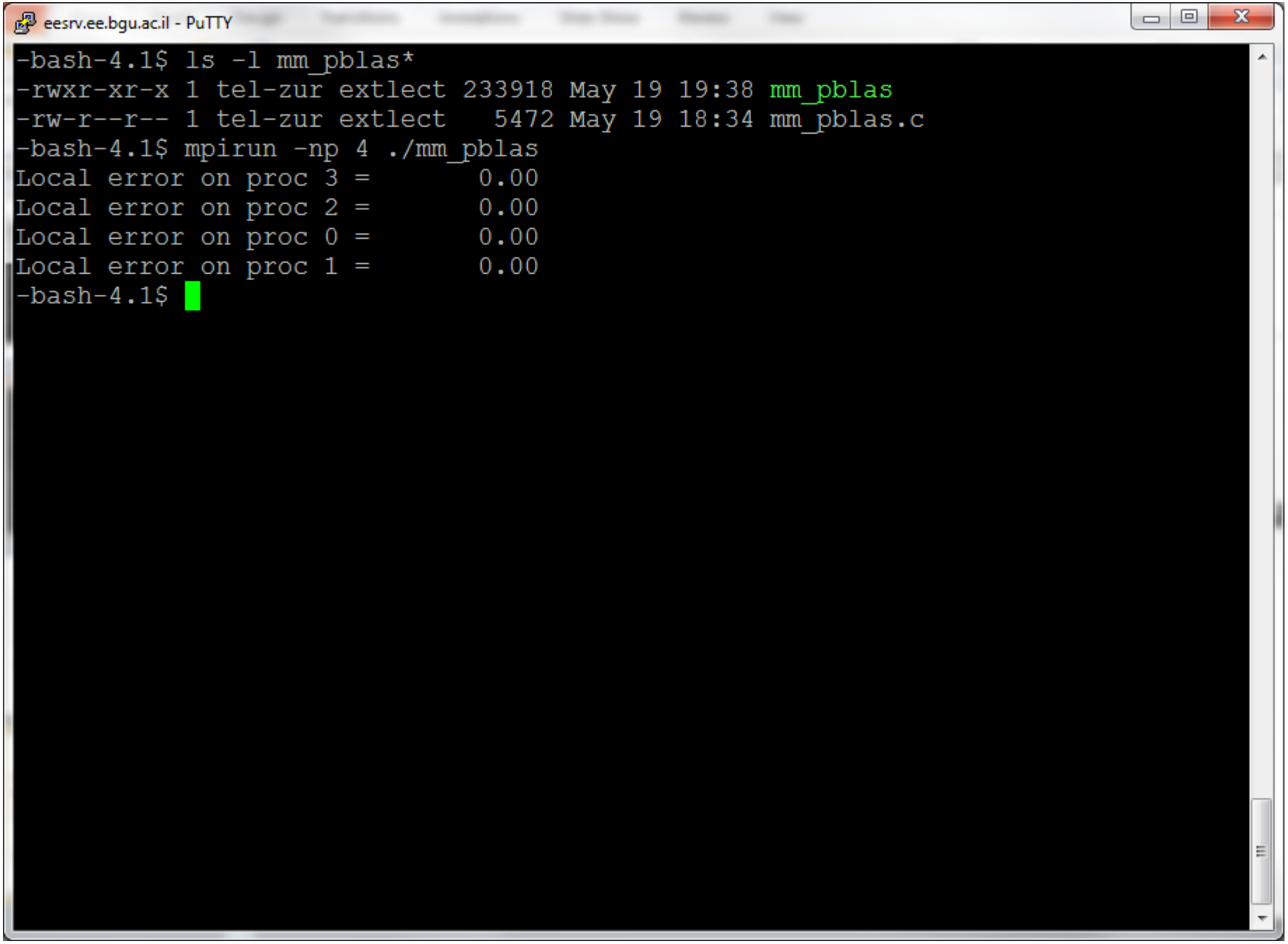
- Block Cyclic Data Distribution Clalculator:

http://acts.nersc.gov/scalapack/hands-on/datadist.html

Compilation on hobbits:

```
mpicc -o mm_pblas mm_pblas.c        \
-L/usr/local/lib -L/usr/lib64/atlas \ -
lscalapack -lptf77blas
```

\ = continuation line


eesrv.ee.bgu.ac.il - PuTTY
-bash-4.1$ ls -l mm_pblas*
-rwxr-xr-x 1 tel-zur extlect 233918 May 19 19:38 mm_pblas
-rw-r--r-- 1 tel-zur extlect 5472 May 19 18:34 mm_pblas.c
-bash-4.1$ mpirun -np 4 ./mm_pblas
Local error on proc 3 = 0.00
Local error on proc 2 = 0.00
Local error on proc 0 = 0.00
Local error on proc 1 = 0.00
-bash-4.1$

mm_pblas on BGU_VM

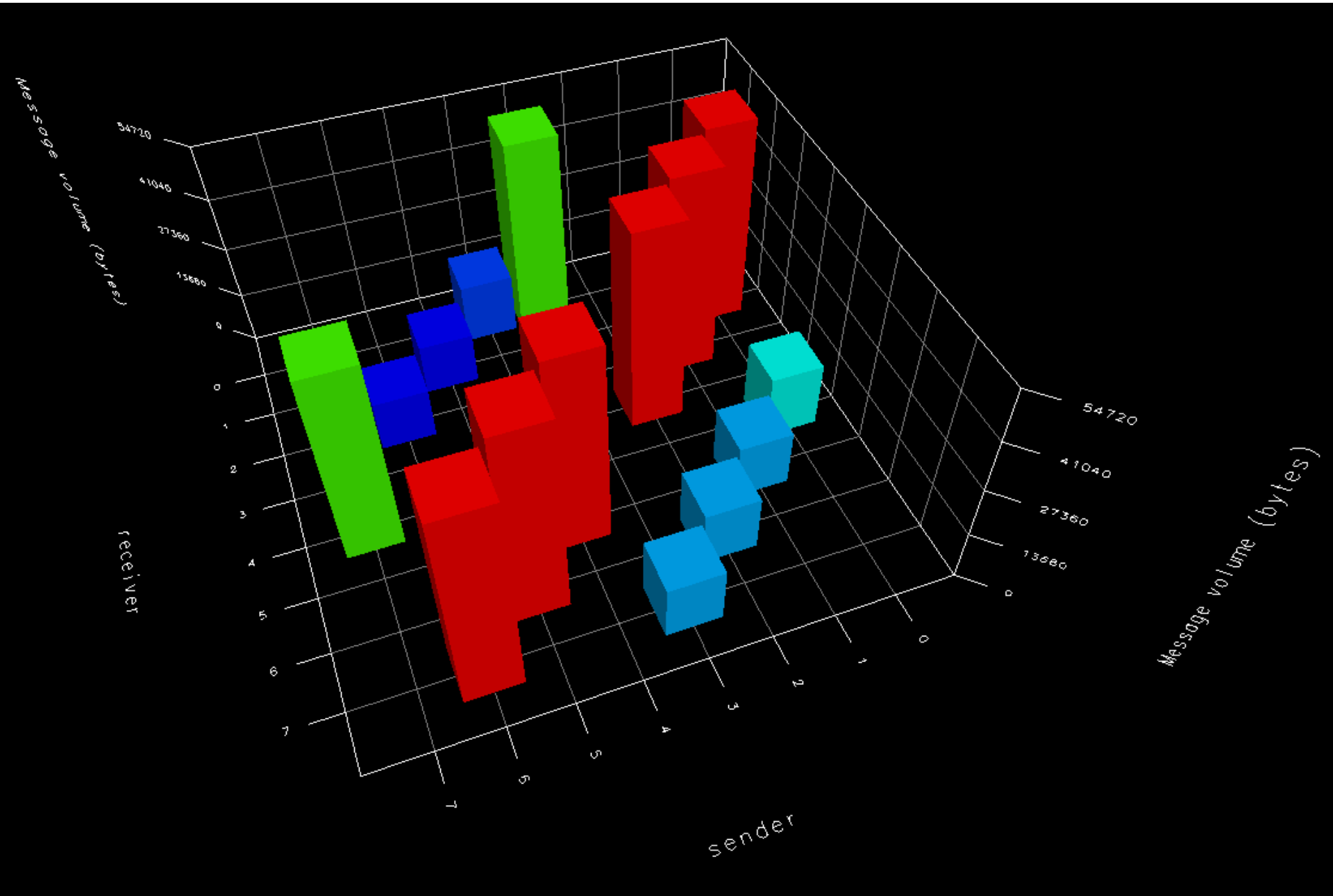
29/11/2020
on BGU_VM
==========

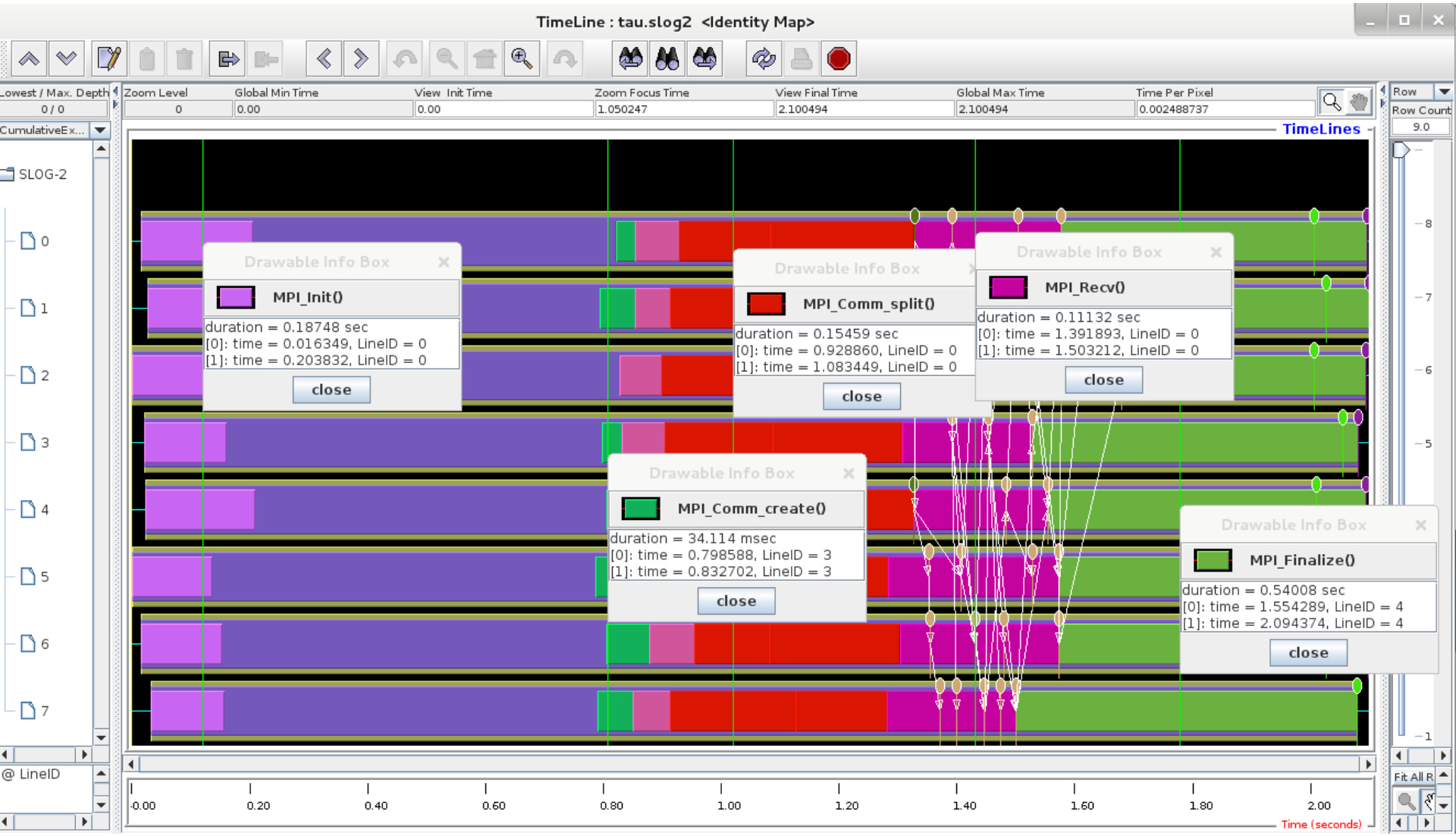
```
sudo yum install scalapack-common --nogpgcheck
sudo yum install scalapack-mpich2 --nogpgchec
sudo yum install scalapack-mpich2-devel --nogpgcheck
sudo yum install scalapack-mpich2-static --nogpgcheck
sudo yum install blacs-common --nogpgcheck
sudo yum install blacs-mpich2 --nogpgcheck
sudo yum install blacs-mpich2-devel --nogpgcheck
sudo yum install blacs-mpich2-static --nogpgchec
sudo yum install lapack-static --nogpgcheck
mpicc -o ./mm_pblas ./mm_pblas.c -L/usr/lib64/mpich2/lib/ -
lscalapack -lmpiblacs -lmpiblacsF77init -L/usr/lib64 -lblas -
llapack
```

In order to use tau_cc.sh add:

```
export LD_LIBRARY_PATH=/usr/lib64/mpich2/lib:
$LD_LIBRARY_PATH
```

mm_pblas


Message volume (bytes)
54720
41040
27360
13680
0
receiver
0
1
2
3
4
5
6
7
sender
0
1
2
3
4
5
6
7
Message volume (bytes)
54720
41040
27360
13680
0


TimeLine : tau.slog2 <Identity Map>
Lowest / Max. Depth
0 / 0
Zoom Level
0
Global Min Time
0.00
View Init Time
0.00
Zoom Focus Time
1.050247
View Final Time
2.100494
Global Max Time
2.100494
Time Per Pixel
0.002488737
Row
Row Count
9.0
TimeLines
SLOG-2
0
1
2
3
4
5
6
7
@ LineID
Drawable Info Box
MPI_Init()
duration = 0.18748 sec
[0]: time = 0.016349, LineID = 0
[1]: time = 0.203832, LineID = 0
close
Drawable Info Box
MPI_Comm_split()
duration = 0.15459 sec
[0]: time = 0.928860, LineID = 0
[1]: time = 1.083449, LineID = 0
close
Drawable Info Box
MPI_Recv()
duration = 0.11132 sec
[0]: time = 1.391893, LineID = 0
[1]: time = 1.503212, LineID = 0
close
Drawable Info Box
MPI_Comm_create()
duration = 34.114 msec
[0]: time = 0.798588, LineID = 3
[1]: time = 0.832702, LineID = 3
close
Drawable Info Box
MPI_Finalize()
duration = 0.54008 sec
[0]: time = 1.554289, LineID = 4
[1]: time = 2.094374, LineID = 4
close
-0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
1.60
1.80
2.00
Time (seconds)
Fit All R

Final remarks

BLAS as dynamic libraries

```
livetau@localhost:/usr/local/ACTS/gnu-4.6.2/openmpi-1.5.4/scalapack-2.0.0
File  Edit  View  Search  Terminal  Help
bash-4.2$ ls /usr/lib64/ | grep blas
libblas.so
libblas.so.3
libblas.so.3.3
libblas.so.3.3.1
bash-4.2$
```

csh> setenv LD_LIBRARY_PATH /path/to/the/libraries/...

bash> export LD_LIBRARY_PATH=/path/to/the/libraries/...

On the hobbit cluster

/usr/lib64/atlas/libcblas.a
/usr/lib64/atlas/libcblas.so
/usr/lib64/atlas/libcblas.so.3
/usr/lib64/atlas/libcblas.so.3.0
/usr/include/cblas.h

hobbit2, 6-10:
/usr/local/lib/libscalapack.a

www.netlib.org/blacs/cblacsqref.ps

Initialization

```
void Cblacs_pinfo   ( int *mypnum, int *nprocs )
void Cblacs_setup   ( int *mypnum, int *nprocs )
void Cblacs_get     ( int icontxt, int what, int *val )
void Cblacs_set     ( int icontxt, int what, int *val )
void Cblacs_gridinit( int *icontxt, char *order,          int nprow, int npcol )
void Cblacs_gridmap ( int *icontxt, int *pmap, int ldpmap, int nprow, int npcol )
```

Destruction

```
void Cblacs_freebuff( int icontxt, int wait )
void Cblacs_gridexit( int icontxt )
void Cblacs_abort   ( int icontxt, int errornum )
void Cblacs_exit    ( int doneflag )
```

Informational and Miscellaneous

```
void Cblacs_gridinfo( int icontxt, int *nprow, int *npcol, int *myprow, int *
int  Cblacs_pnum    ( int icontxt,          int  prow, int  pcol )
void Cblacs_pcoord  ( int icontxt, int pnum, int *prow, int *pcol )
void Cblacs_barrier( int icontxt, char *scope )
```

Non-standard

```
void   Csetpvmtids ( int ntasks, int *tids )
double Cdcputime00 ( )
double Cdwalltime00( )
int    Cksendid    ( int icontxt,             int rdest, int cdest )
int    Ckrecvid    ( int icontxt,             int rsrc,  int csrc  )
int    Ckbsid      ( int icontxt, char *scope                      )
int    Ckbrid      ( int icontxt, char *scope, int rsrc,  int csrc  )
```

Sending

```
void C□gesd2d( int icontxt,                                    int m, int n, TYPE *A, int lda, int rdest, int cdest )

void C□gebs2d( int icontxt, char *scope, char *top,            int m, int n, TYPE *A, int lda )

void C□trsd2d( int icontxt, char *uplo, char *diag,            int m, int n, TYPE *A, int lda, int rdest, int cdest )

void C□trbs2d( int icontxt, char *scope, char *top, char *uplo, char *diag, int m, int n, TYPE *A, int lda)
```

Receiving

```
void C□gerv2d( int icontxt,                                    int m, int n, TYPE *A, int lda, int rsrc, int csrc )

void C□gebr2d( int icontxt, char *scope, char *top,            int m, int n, TYPE *A, int lda, int rsrc, int csrc )

void C□trrv2d( int icontxt, char *uplo, char *diag,            int m, int n, TYPE *A, int lda, int rsrc, int csrc )

void C□trbr2d( int icontxt, char *scope, char *top, char *uplo, char *diag, int m, int n, TYPE *A, int lda, int rsrc, int csrc )
```

Combine Operations

```
void C□gamx2d( int icontxt, char *scope, char *top, int m, int n, TYPE *A, int lda, int *RA, int *CA, int RCflag, int rdest, int cdest )

void C□gamn2d( int icontxt, char *scope, char *top, int m, int n, TYPE *A, int lda, int *RA, int *CA, int RCflag, int rdest, int cdest )

void C□gsum2d( int icontxt, char *scope, char *top, int m, int n, TYPE *A, int lda,                           int rdest, int cdest )
```

Definition of □

□is	Data operated on is	TYPE is
s	single precision real	float
d	double precision real	double
c	single precision complex	float
z	double precision complex	double
i	integer	int

Options

```
UPLO  = "Upper triangular", "Lower triangular";
DIAG  = "Non-unit triangular", "Unit triangular";
SCOPE = "All", "row", "column";
TOP   = (SEE DESCRIPTION BELOW).
```

Level 1 BLAS

	dim	scalar	vector	vector	scalars		5-element array		prefixes
SUBROUTINE xROTG (						A, B, C, S)		Generate plane rotation	S, D
SUBROUTINE xROTMG(					D1, D2,	A, B,	PARAM)	Generate modified plane rotation	S, D
SUBROUTINE xROT (	N,		X, INCX,	Y, INCY,		C, S)		Apply plane rotation	S, D
SUBROUTINE xROTM (	N,		X, INCX,	Y, INCY,			PARAM)	Apply modified plane rotation	S, D
SUBROUTINE xSWAP (	N,		X, INCX,	Y, INCY)				$x \leftrightarrow y$	S, D, C, Z
SUBROUTINE xSCAL (	N,	ALPHA,	X, INCX)					$x \leftarrow \alpha x$	S, D, C, Z, CS, ZD
SUBROUTINE xCOPY (	N,		X, INCX,	Y, INCY)				$y \leftarrow x$	S, D, C, Z
SUBROUTINE xAXPY (	N,	ALPHA,	X, INCX,	Y, INCY)				$y \leftarrow \alpha x + y$	S, D, C, Z
FUNCTION xDOT (	N,		X, INCX,	Y, INCY)				$dot \leftarrow x^T y$	S, D, DS
FUNCTION xDOTU (	N,		X, INCX,	Y, INCY)				$dot \leftarrow x^T y$	C, Z
FUNCTION xDOTC (	N,		X, INCX,	Y, INCY)				$dot \leftarrow x^H y$	C, Z
FUNCTION xxDOT (	N,		X, INCX,	Y, INCY)				$dot \leftarrow \alpha + x^T y$	SDS
FUNCTION xNRM2 (	N,		X, INCX)					$nrm2 \leftarrow \|\|x\|\|_2$	S, D, SC, DZ
FUNCTION xASUM (	N,		X, INCX)					$asum \leftarrow \|\|re(x)\|\|_1 + \|\|im(x)\|\|_1$	S, D, SC, DZ
FUNCTION IxAMAX(	N,		X, INCX)					$amax \leftarrow 1^{st} k \ni \|re(x_k)\| + \|im(x_k)\| = max(\|re(x_i)\| + \|im(x_i)\|)$	S, D, C, Z

Level 2 BLAS

	options			dim	b-width	scalar	matrix	vector	scalar	vector		
xGEMV (		TRANS,		M, N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xGBMV (		TRANS,		M, N,	KL, KU,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xHEMV (	UPLO,			N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHBMV (	UPLO,			N, K,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHPMV (	UPLO,			N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xSYMV (	UPLO,			N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSBMV (	UPLO,			N, K,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSPMV (	UPLO,			N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xTRMV (	UPLO,	TRANS,	DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTBMV (	UPLO,	TRANS,	DIAG,	N, K,			A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTPMV (	UPLO,	TRANS,	DIAG,	N,			AP,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTRSV (	UPLO,	TRANS,	DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z
xTBSV (	UPLO,	TRANS,	DIAG,	N, K,			A, LDA,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z
xTPSV (	UPLO,	TRANS,	DIAG,	N,			AP,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z

	options			dim	scalar	vector	vector	matrix		
xGER (				M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + A, A - m \times n$	S, D
xGERU (				M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + A, A - m \times n$	C, Z
xGERC (				M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^H + A, A - m \times n$	C, Z
xHER (	UPLO,			N,	ALPHA,	X, INCX,		A, LDA)	$A \leftarrow \alpha xx^H + A$	C, Z
xHPR (	UPLO,			N,	ALPHA,	X, INCX,		AP)	$A \leftarrow \alpha xx^H + A$	C, Z
xHER2 (	UPLO,			N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^H + y(\alpha x)^H + A$	C, Z
xHPR2 (	UPLO,			N,	ALPHA,	X, INCX,	Y, INCY,	AP)	$A \leftarrow \alpha xy^H + y(\alpha x)^H + A$	C, Z
xSYR (	UPLO,			N,	ALPHA,	X, INCX,		A, LDA)	$A \leftarrow \alpha xx^T + A$	S, D
xSPR (	UPLO,			N,	ALPHA,	X, INCX,		AP)	$A \leftarrow \alpha xx^T + A$	S, D
xSYR2 (	UPLO,			N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + \alpha yx^T + A$	S, D
xSPR2 (	UPLO,			N,	ALPHA,	X, INCX,	Y, INCY,	AP)	$A \leftarrow \alpha xy^T + \alpha yx^T + A$	S, D

Level 3 BLAS

	options				dim		scalar	matrix	matrix	scalar	matrix		
xGEMM (			TRANSA,	TRANSB,	M, N,	K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha op(A)op(B) + \beta C, op(X) = X, X^T, X^H, C - m \times n$	S, D, C, Z
xSYMM (	SIDE,	UPLO,			M, N,		ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^T$	S, D, C, Z
xHEMM (	SIDE,	UPLO,			M, N,		ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^H$	C, Z
xSYRK (		UPLO,	TRANS,		N,	K,	ALPHA,	A, LDA,		BETA,	C, LDC)	$C \leftarrow \alpha AA^T + \beta C, C \leftarrow \alpha A^T A + \beta C, C - n \times n$	S, D, C, Z
xHERK (		UPLO,	TRANS,		N,	K,	ALPHA,	A, LDA,		BETA,	C, LDC)	$C \leftarrow \alpha AA^H + \beta C, C \leftarrow \alpha A^H A + \beta C, C - n \times n$	C, Z
xSYR2K(		UPLO,	TRANS,		N,	K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB^T + \bar{\alpha} BA^T + \beta C, C \leftarrow \alpha A^T B + \bar{\alpha} B^T A + \beta C, C - n \times n$	S, D, C, Z
xHER2K(		UPLO,	TRANS,		N,	K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB^H + \bar{\alpha} BA^H + \beta C, C \leftarrow \alpha A^H B + \bar{\alpha} B^H A + \beta C, C - n \times n$	C, Z
xTRMM (	SIDE,	UPLO,	TRANSA,	DIAG,	M, N,		ALPHA,	A, LDA,	B, LDB)			$B \leftarrow \alpha op(A)B, B \leftarrow \alpha Bop(A), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z
xTRSM (	SIDE,	UPLO,	TRANSA,	DIAG,	M, N,		ALPHA,	A, LDA,	B, LDB)			$B \leftarrow \alpha op(A^{-1})B, B \leftarrow \alpha Bop(A^{-1}), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z

Meaning of prefixes

S - `REAL`	C - `COMPLEX`
D - `DOUBLE PRECISION`	Z - `COMPLEX*16` (may not be supported by all machines)

Level 2 and Level 3 PBLAS Matrix Types

GE - GEneral
SY - SYmmetric
HE - HErmitian
TR - TRiangular

Level 2 and Level 3 PBLAS Options

Dummy options arguments are declared as CHARACTER*1 and may be passed as character strings.
TRANS□ = 'No transpose', 'Transpose', 'Conjugate transpose', (X, X^T, X^H)
UPLO = 'Upper triangular', 'Lower Triangular'
DIAG = 'Non-unit triangular', 'Unit triangular'
SIDE = 'Left' or 'Right' (A or op(A) on the left, or A or op(A) on the right)

For real matrices, TRANS□ = 'T' and TRANS□ = 'C' have the same meaning.

For Hermitian matrices, TRANS□='T' is not allowed.

For complex symmetric matrices, TRANS□='C' is not allowed.

Obtaining the software via netlib

In order to get instructions for downloading the PBLAS, send email to `netlib@ornl.gov` and in the body of the message type `send index from scalapack`.

Send comments, questions to `scalapack@cs.utk.edu`.

Array Descriptor, Increment

The array descriptor *DESCA* is an integer array of dimension 9. It describes the two-dimensional block-cyclic mapping of the matrix A.

The first two entries are the descriptor type and the BLACS context. The third and fourth entries are the dimensions of the matrix (row, column). The fifth and sixth entries are the row- and column block sizes used to distribute the matrix. The seventh and eighth are the coordinates of the process containing the first entry of the matrix. The last entry contains the leading dimension of the local array containing the matrix elements.

The increment specified for vectors is always global. So far only 1 and DESCA(M_) are supported.

References

J. Dongarra and R. C. Whaley, LAPACK, Working Note 94, *A User's Guide to the BLACS v1.0*, Computer Science Dept. Technical Report CS-95-281, University of Tennessee, Knoxville, March, 1995. To receive a postscript copy, send email to netlib@ornl.gov and in the mail message type: `send lawn94.ps from lapack/lawns`.

J. Choi, J. Dongarra, and D. Walker, *PB-BLAS: A Set of Parallel Block Basic Linear Algebra Subroutines*, Proceedings of Scalable High Performance Computing Conference (Knoxville, TN), pp. 534-541, IEEE Computer Society Press, May 23-25, 1994.

J. Choi, J. Demmel, I. Dhillon, J. Dongarra, S. Ostrouchov, A. Petitet, K. Stanley, D. Walker and R. C. Whaley, LAPACK, Working Note 95, *ScaLAPACK: A Scalable Linear Algebra library for Distributed Memory Concurrent Computers - Design Issues and Performance*, Computer Science Dept. Technical Report CS-95-283, University of Tennessee, Knoxville, March 1995. To receive a postscript copy, send email to netlib@ornl.gov and in the mail message type: `send lawn95.ps from lapack/lawns`.

J. Choi, J. Dongarra, S. Ostrouchov, A. Petitet, D. Walker and R. C. Whaley, LAPACK, Working Note 100, *A Proposal for a Set of Parallel Basic Linear Algebra Subprograms*, Computer Science Dept. Technical Report CS-95-292, University of Tennessee, Knoxville, July 1995. To receive a postscript copy, send email to netlib@ornl.gov and in the mail message type: `send lawn100.ps from lapack/lawns`.

Parallel Basic Linear Algebra Subprograms

Release 1.0

University of Tennessee

March 28, 1995

A Quick Reference Guide

http://www.netlib.org/scalapack/pblas_qref.html

Level 1 PBLAS

	dim scalar	vector	vector		prefixes
P□SWAP	(N,	X, IX, JX, DESCX, INCX,	Y, IY, JY, DESCY, INCY)	$x \leftrightarrow y$	S, D, C, Z
P□SCAL	(N, ALPHA,	X, IX, JX, DESCX, INCX)		$x \leftarrow \alpha x$	S, D, C, Z, CS, ZD
P□COPY	(N,	X, IX, JX, DESCX, INCX,	Y, IY, JY, DESCY, INCY)	$y \leftarrow x$	S, D, C, Z
P□AXPY	(N, ALPHA,	X, IX, JX, DESCX, INCX,	Y, IY, JY, DESCY, INCY)	$y \leftarrow \alpha x + y$	S, D, C, Z
P□DOT	(N, DOT,	X, IX, JX, DESCX, INCX,	Y, IY, JY, DESCY, INCY)	$dot \leftarrow x^T y$	S, D
P□DOTU	(N, DOTU,	X, IX, JX, DESCX, INCX,	Y, IY, JY, DESCY, INCY)	$dotu \leftarrow x^T y$	C, Z
P□DOTC	(N, DOTC,	X, IX, JX, DESCX, INCX,	Y, IY, JY, DESCY, INCY)	$dotc \leftarrow x^H y$	C, Z
P□NRM2	(N, NORM2,	X, IX, JX, DESCX, INCX)		$norm2 \leftarrow \|x\|_2$	S, D, SC, DZ
P□ASUM	(N, ASUM,	X, IX, JX, DESCX, INCX)		$asum \leftarrow \|re(x)\|_1 + \|im(x)\|_1$	S, D, SC, DZ
P□AMAX	(N, AMAX, INDX,	X, IX, JX, DESCX, INCX)		$indx \leftarrow 1^{st}\ k \ni \lvert Re(x_k)\rvert + \lvert Im(x_k)\rvert = max(\lvert Re(x_i)\rvert + \lvert Im(x_i)\rvert) = amax$	S, D, C, Z

Level 2 PBLAS

	options	dim	scalar	matrix		vector		scalar	vector		
P□GEMV	(TRANS,	M, N,	ALPHA,	A, IA, JA, DESCA,	X, IX, JX, DESCX,	INCX,	BETA,	Y, IY, JY, DESCY,	INCY)	$y \leftarrow \alpha op(A)x + \beta y, op(A) = A, A^T, A^H, A - m \times n$	S, D, C, Z
P□HEMV	(UPLO,	N,	ALPHA,	A, IA, JA, DESCA,	X, IX, JX, DESCX,	INCX,	BETA,	Y, IY, JY, DESCY,	INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
P□SYMV	(UPLO,	N,	ALPHA,	A, IA, JA, DESCA,	X, IX, JX, DESCX,	INCX,	BETA,	Y, IY, JY, DESCY,	INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
P□TRMV	(UPLO, TRANS, DIAG,	N,		A, IA, JA, DESCA,	X, IX, JX, DESCX,	INCX)				$x \leftarrow \alpha Ax, x \leftarrow \alpha A^T x, x \leftarrow \alpha A^H x,$	S, D, C, Z
P□TRSV	(UPLO, TRANS, DIAG,	N,		A, IA, JA, DESCA,	X, IX, JX, DESCX,	INCX)				$x \leftarrow \alpha A^{-1} x, x \leftarrow \alpha A^{-T} x, x \leftarrow \alpha A^{-H} x,$	S, D, C, Z

	options	dim	scalar	vector		vector		matrix			
P□GER	(	M, N,	ALPHA,	X, IX, JX, DESCX,	INCX,	Y, IY, JY, DESCY,	INCY,	A, IA, JA, DESCA)		$A \leftarrow \alpha x y^T + A, A - m \times n$	S, D
P□GERU	(	M, N,	ALPHA,	X, IX, JX, DESCX,	INCX,	Y, IY, JY, DESCY,	INCY,	A, IA, JA, DESCA)		$A \leftarrow \alpha x y^T + A, A - m \times n$	C, Z
P□GERC	(	M, N,	ALPHA,	X, IX, JX, DESCX,	INCX,	Y, IY, JY, DESCY,	INCY,	A, IA, JA, DESCA)		$A \leftarrow \alpha x y^H + A, A - m \times n$	C, Z
P□HER	(UPLO,	N,	ALPHA,	X, IX, JX, DESCX,	INCX,			A, IA, JA, DESCA)		$A \leftarrow \alpha x x^H + A$	C, Z
P□HER2	(UPLO,	N,	ALPHA,	X, IX, JX, DESCX,	INCX,	Y, IY, JY, DESCY,	INCY,	A, IA, JA, DESCA)		$A \leftarrow \alpha x y^H + y(\alpha x)^H + A$	C, Z
P□SYR	(UPLO,	N,	ALPHA,	X, IX, JX, DESCX,	INCX,			A, IA, JA, DESCA)		$A \leftarrow \alpha x x^T + A$	S, D
P□SYR2	(UPLO,	N,	ALPHA,	X, IX, JX, DESCX,	INCX,	Y, IY, JY, DESCY,	INCY,	A, IA, JA, DESCA)		$A \leftarrow \alpha x y^T + \alpha y x^T + A$	S, D

Level 3 PBLAS

	options		dim	scalar	matrix			matrix			scalar	matrix		
P□GEMM	(	TRANSA, TRANSB,	M, N, K,	ALPHA,	A, IA, JA, DESCA,	B, IB, JB, DESCB,					BETA,	C, IC, JC, DESCC)	$C \leftarrow \alpha op(A)op(B) + \beta C, op(X) = X, X^T, X^H, C - m \times n$	S, D, C, Z
P□SYMM	(SIDE, UPLO,		M, N,		ALPHA,	A, IA, JA, DESCA,	B, IB, JB, DESCB,				BETA,	C, IC, JC, DESCC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^T$	S, D, C, Z
P□HEMM	(SIDE, UPLO,		M, N,		ALPHA,	A, IA, JA, DESCA,	B, IB, JB, DESCB,				BETA,	C, IC, JC, DESCC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^H$	C, Z
P□SYRK	(	UPLO, TRANS,		N, K,	ALPHA,	A, IA, JA, DESCA,					BETA,	C, IC, JC, DESCC)	$C \leftarrow \alpha AA^T + \beta C, C \leftarrow \alpha A^T A + \beta C, C - n \times n$	S, D, C, Z
P□HERK	(	UPLO, TRANS,		N, K,	ALPHA,	A, IA, JA, DESCA,					BETA,	C, IC, JC, DESCC)	$C \leftarrow \alpha AA^H + \beta C, C \leftarrow \alpha A^H A + \beta C, C - n \times n$	C, Z
P□SYR2K	(	UPLO, TRANS,		N, K,	ALPHA,	A, IA, JA, DESCA,	B, IB, JB, DESCB,				BETA,	C, IC, JC, DESCC)	$C \leftarrow \alpha AB^T + \alpha BA^T + \beta C, C \leftarrow \alpha A^T B + \alpha B^T A + \beta C, C - n \times n$	S, D, C, Z
P□HER2K	(	UPLO, TRANS,		N, K,	ALPHA,	A, IA, JA, DESCA,	B, IB, JB, DESCB,				BETA,	C, IC, JC, DESCC)	$C \leftarrow \alpha AB^H + \bar{\alpha} BA^H + \beta C, C \leftarrow \alpha A^H B + \bar{\alpha} B^H A + \beta C, C - n \times n$	C, Z
P□TRAN	(		M, N,		ALPHA,	A, IA, JA, DESCA,					BETA,	C, IC, JC, DESCC)	$C \leftarrow \beta C + \alpha A^T, A - n \times m, C - m \times n$	S, D
P□TRANU	(		M, N,		ALPHA,	A, IA, JA, DESCA,					BETA,	C, IC, JC, DESCC)	$C \leftarrow \beta C + \alpha A^T, A - n \times m, C - m \times n$	C, Z
P□TRANC	(		M, N,		ALPHA,	A, IA, JA, DESCA,					BETA,	C, IC, JC, DESCC)	$C \leftarrow \beta C + \alpha A^H, A - n \times m, C - m \times n$	C, Z
P□TRMM	(SIDE, UPLO, TRANSA,	DIAG,	M, N,		ALPHA,	A, IA, JA, DESCA,	B, IB, JB, DESCB)						$B \leftarrow \alpha op(A)B, B \leftarrow \alpha Bop(A), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z
P□TRSM	(SIDE, UPLO, TRANSA,	DIAG,	M, N,		ALPHA,	A, IA, JA, DESCA,	B, IB, JB, DESCB)						$B \leftarrow \alpha op(A^{-1})B, B \leftarrow \alpha Bop(A^{-1}), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z