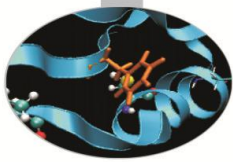
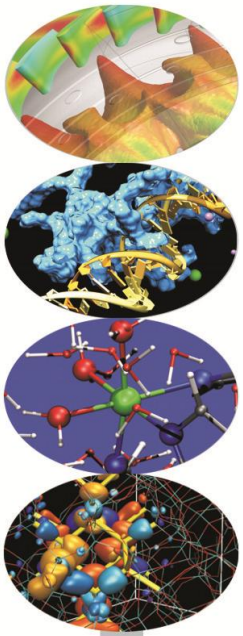
CINECA SCAI
SuperComputing Applications and Innovation





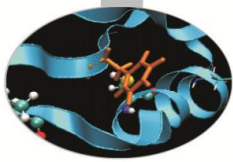
PETSc Exercises

Simone Bnà – s.bna@cineca.it
SuperComputing Applications and Innovation Department

PRACE

CINECA

1 Exercise: 1_Petsc_hello.c



Write a PETSc program where:

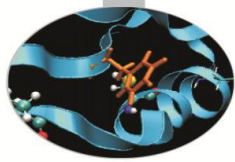
- one processor prints on the std out the number of processors and its rank
- All the processors print the string “Hello by proc <rank_of the_processor>!”

A template of the program with some hints and the solution can be found here: https://github.com/sbna/CINECA-SCAI-HPC-courses/

You can edit the makefile included in the src files or generate it with cmake.

All the instructions on how to compile and run the examples can be found in the README of the petsc folder.

2 Exercise: 2_Petsc_vec.c



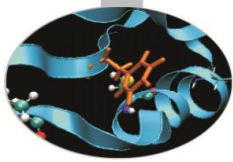
Write a PETSc program that creates a parallel vector and fills it according to the formula

$$v[i] = i$$

in the different ways:

- Each processor fills all the entries
- Each processor fills only its local entries using VecSetValue
- Each processor fills only its local entries using VecGetArray
- Each processor prints the global and local size of the vector

3 Exercise: 3_Petsc_mat.c

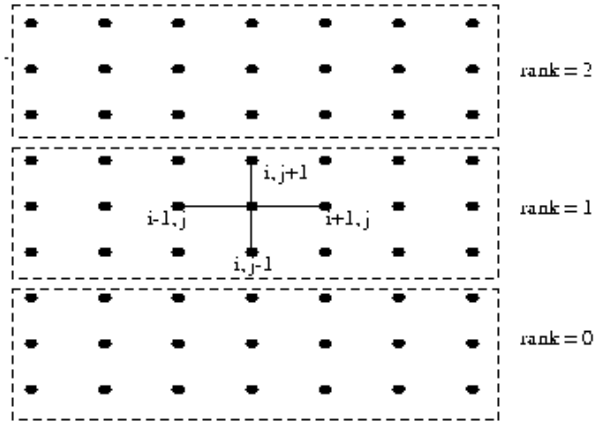


Write a PETSc program that creates a parallel matrix to host the 2d (5 point stencil) finite difference discretization of the Laplace operator. Remember that

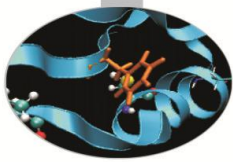
- Each processor needs to insert only elements that it owns locally (but any non-local elements will be sent to the appropriate processor during matrix assembly)
- Each processor prints the global and local size of the matrix

Hints:

$$\Delta f(x,y) \approx \frac{f(x-h,y)+f(x+h,y)+f(x,y-h)+f(x,y+h)-4f(x,y)}{h^2}.$$



3 Exercise: 3_Petsc_mat.c



If we use the natural ordering that orders first all the unknowns for x = h then x = 2h etc, the following snippet of code can be used:

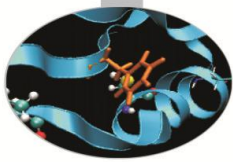
```
/*
0<i<m, m=number of point in the x direction
0<j<n  n=number of points in the y direction

[Istart, Iend] is the range of rows of the matrix locally owned
*/

Mat  A;
PetscInt  i,j,Ii,J,Istart,Iend,m=8,n=8;
PetscScalar  v;

for (Ii=Istart; Ii<Iend; Ii++) {
  v = -1.0; i = Ii/n; j = Ii - i*n;
  if (i>0)   {J = Ii - n; MatSetValues(A,1,&Ii,1,&J,&v,ADD_VALUES);}
  if (i<m-1) {J = Ii + n; MatSetValues(A,1,&Ii,1,&J,&v,ADD_VALUES);}
  if (j>0)   {J = Ii - 1; MatSetValues(A,1,&Ii,1,&J,&v,ADD_VALUES);}
  if (j<n-1) {J = Ii + 1; MatSetValues(A,1,&Ii,1,&J,&v,ADD_VALUES);}
  v = 4.0; MatSetValues(A,1,&Ii,1,&Ii,&v,ADD_VALUES);
}
```

4 Exercise: 4_Petsc_ksp_poisson.c



Write a PETSc program that solve in parallel the following PDE:

$$- \nabla^2 u = -32 * (x * (x-1) + y * (y-1)) \quad \text{in} \quad \Omega = [0,1][0,1]$$
$$\text{u} = 0 \text{ in x=0, x=1, y=0, y=1}$$

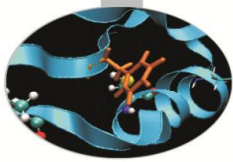
Note that the boundary conditions of the problem are Dirichlet in all the edges of the square.

The analytical solution is:

$$u = 16 * x * (x-1) * y * (y-1)$$

- Use a distributed array (DMDA) to manage the parallel structured mesh (with uniform coordinates) and vectors.
- Solve the problem with a KSP object (default one is GMRES preconditioned by ILU).
- Check if the numerical solution is close in l2 norm to the analytical solution.

5 Exercise: 5_Petsc_snes_poisson.c



Write a PETSc program that solve in parallel the same PDE as before:

$$- \nabla^2 u = -32 * (x * (x-1) + y * (y-1)) \quad \text{in} \quad \Omega = [0,1][0,1]$$
$$\text{u} = 0 \text{ in x=0, x=1, y=0, y=1}$$

- Use a distributed array (DMDA) to manage the parallel structured mesh (with uniform coordinates) and vectors.
- Solve the problem with a SNES object providing the following two functions:
  - PetscErrorCode MyComputeFunction(SNES,Vec,Vec,void*);
  - PetscErrorCode MyComputeJacobian(SNES,Vec,Mat,Mat,void*);