Introduction to PETSc: Matrix-Free Matrices

Serge Van Criekingen

Maison de la Simulation

14 mai 2013

PETSc Matrix-Free Matrix in C

```
extern PetscErrorCode myMatMult(Mat,Vec,Vec);
...
MatCreateShell(comm,m,n,M,N,null,&theShellMatrix);
MatShellSetOperation(theShellMatrix,MATOP_MULT,
                                (void(*)(void))myMatMult);

                    ⋮

PetscErrorCode myMatMult(Mat theShellMatrix,Vec T,Vec AT)
{
...
}
```

PETSc Matrix-Free Matrix in Fortran

In Fortran :

```
external myMatMult
...
call MatCreateShell(comm,m,n,M,N,PETSC_NULL_INTEGER,
                                          theShellMatrix,ierr)
call MatShellSetOperation(theShellMatrix,MATOP_MULT,
                                             myMatMult,ierr)

                    :


subroutine myMatMult(theShellMatrix,T,AT,ierr)
...
end subroutine myMatMult
```

PETSc Matrix-Free Matrix *with context* in C

```
extern PetscErrorCode myMatMult(Mat,Vec,Vec);
...
ContextType ctx;
ctx = ...;
MatCreateShell(comm,m,n,M,N,(void*)&ctx,&theShellMatrix);
MatShellSetOperation(theShellMatrix,MATOP_MULT,
                                (void(*)(void))myMatMult);

                    ⋮

PetscErrorCode myMatMult(Mat theShellMatrix,Vec T,Vec AT)
{
   ContextType* ctx_ptr;
   MatShellGetContext(theShellMatrix, (void**) &ctx_ptr);
   ...
}
```

Matrix-Free Matrix : Exercice

Implement a matrix-free product corresponding to the diagonal matrix with the value 2. on each diagonal element.

Apply to a parallel vector of your choice and check the result.

Use a context to attach the diagonal value to the shell matrix, and do the same.

Matrix-Free Matrix : Fortran solution to exercise

```
program matrixFree
 ...
 call MatCreateShell(PETSC_COMM_WORLD,PETSC_DECIDE,PETSC_DECIDE,
             globalSize,globSize,PETSC_NULL_INTEGER,theShellMatrix,ierr)
 call MatShellSetOperation(theShellMatrix,MATOP_MULT,myMatMult,ierr)
 call VecCreateMPI(PETSC_COMM_WORLD,PETSC_DECIDE,globSize,theVecIn,ierr)
 val = 3.0
 call VecSet(theVecIn,val,ierr)
 call VecAssemblyBegin(theVecIn,ierr)
 call VecAssemblyEnd(theVecIn,ierr)
 call VecDuplicate(theVecIn,theVecOut,ierr)
 call MatMult(theShellMatrix,theVecIn,theVecOut,ierr)
 ...
end program matrixFree

subroutine myMatMult(theShellMatrix,T,AT,ierr)
 ...
 call VecCopy(T,AT,ierr)
 factor = 2
 call VecScale (AT,factor,ierr)
 call VecAssemblyBegin(AT,ierr)
 call VecAssemblyEnd(AT,ierr)
 ...
end subroutine myMatMult
```