# EllPETSc

Paul Walker, Gerd Lanfermann

Date: 2002/06/04 12:51:09

**Abstract**

`EllPETSc` provides 3D elliptic solvers for the various classes of elliptic problems defined in `EllBase`. `EllPETSc` using the "Portable, Extendable Toolkit for Scientific computation" (PETSc) by Argonne National Lab. PETSc is a suite of routines and data structures that can be employed for solving partial differential equations in parallel. `EllPETSc` t is called by the interfaces provided in `EllBase`.

## 1 Purpose

This thorns provides sophisticated solvers based on PETSc libraries. It supports all the interfaces defined in `EllBase`. At this point is not optimized for performance. Expect improvements as we develop the elliptic solver arrangement.

This thorn provides

1. No Pizza
2. No Wine
3. peace

## 2 Technical Details

This thorn supports three elliptic problem classes: **LinFlat** for a standard 3D cartesian Laplace operator, using the standard 7-point computational molecule. **LinMetric** for a Laplace operator derived from the metric, using 19-point stencil. **LinConfMetric** for a Laplace operator derived from the metric and a conformal factor, using a 19-point stencil. The code of the solvers differs for the classes and is explained in the following section.

### 2.1 Installing PETSc

PETSc needs to be installed on the machine and the environment variables `PETSC_ARCH` and `PETSC_DIR` have to be set to compile `EllPETSc`. PETSc can obtained for free at `http://www-fp.mcs.anl.gov/petsc/`. Cactus needs to be compiled with MPI. While PETSc can be compiled for single processor mode (without MPI), Cactus has only been tested and used with the parallel version of PETSc requiring MPI. For detailed information in how to install PETSc refer to the documentation.

### 2.2 LinFlat

For this class we employ the the 7-point stencil based on only. These values are constant at each gridpoint.

### 2.3 LinMetric

For this class the standard 19-point stencil is initialized, taken the underlying metric into account. The values for the stencil function differ at each gridpoints.

### 2.4 LinConfMetric

For this class the standard 19-point stencil is initialized, taken the underlying metric and its conformal factor into account. The values for the stencil function differ at each gridpoints.

### 2.5 Interfacing PETSc

The main task when interfacing PETSc consists of transferring data from the Cactus parallel data structures (gridfunctions) to the parallel structures provided by PETSc.

Here we explain the main steps, to be read with the code at hands.

1. The indices `imin,imax ...` are calculated and describe the starting/ending points in *3D local index space*: ghostzones are not included here.
2. A *linear global index* is calculated describing the starting/ending points in *linear global index space*. Ghostzone are not included here.
3. A lookup gridfunction `wsp` is loaded identifying the *3D local index* with the *linear global index*. Values of zero indicate boundaries.
4. PETSc matrices/vectors are created specifying the linear size: global endpoint - global startpoint.
5. For the elliptic class `LinFlat` the stencil functions are initialized with the standard 7-point stencil, the class `LinMetric` and `LinConfMetric` require a more sophisticated treatment described later.
6. Looping over the processor local grid points (in 3D local index space) the PETSc vectors and coefficient matrix is loaded if no boundary is present (`wsp[i,j,k]` not equal zero.);
7. Starting the PETSc vector and matrix assembly, nested for performance as recommended by PETSc.
8. Creation of the elliptic solver context and setting of options, followed by the call to the PETSc solver.
9. Upon completion of the solve, the PETSc solution has to transferred to the Cactus data structures.

## 3 Comments

The sizes of the arrays `Mlinear` for the coefficient matrix and `Nsource` are passed in the solver. A storage flag is set if these variables are of a sized greater 1. In this case, the array can be accessed.

## 4 General remarks: PETSc within Cactus

### 4.1 PETSc in src code

Use PETSc as normal, Use the PUGH communicator if a routine needs a communictor. On first pass, you need to make a call to PETScSetCommWorld() and PetscInitialize() to set the PETSc communicator and initialize PETSc.

This could be a seperate routine scheduled early in schedule.ccl at BASEGRID eg. PetscInitialize() requires the commandline parameters as input. It allows you to pass through the flags, etc. (I have not ried this feature.) Initialize the PETSc communicator with the Cactus communicator. You end up having code like this:

```
/* The pugh Extension handle */
pGH *pughGH;

/* Get the link to pugh Extension */
pughGH = (pGH*)GH->extensions[CCTK_GHExtensionHandle("PUGH")];
```

```
  if (first_trip==0)
  {
    int argc;
    char **argv;

    /* Get the commandline arguments */
    argc = CCTK_CommandLine(&argv);

    /* Set the PETSc communicator to set of
       PUGH and initialize PETSc */
    ierr = PetscSetCommWorld(pughGH->PUGH_COMM_WORLD); CHKERRA(ierr);
    PetscInitialize(&argc,&argv,NULL,NULL);

    CCTK_INFO("PETSc initialized");
  }
```

## 4.2 make.code.defn

You need to tell Cactus to look for the PETSc includes: In the file make.code.defn define the SRCS (sources) as explained in the dcoumentation and add a lien for SUS_INC_DIR which lets Cactus look for additional includes, eg.:

```
SYS_INC_DIRS += $(PETSC_DIR) $(PETSC_DIR)/include  \
                $(PETSC_DIR)/bmake/$(PETSC_ARCH)
```

## 4.3 make.configuration.defn

This file is not created by the when you use Cactus to create a new thorn by "gmake newthorn". For a template PETSc configuration file, have a look in ./CactusElliptic/EllPETSc/src/make.configuration.defn.

The first section checks if PETSC_DIR/PETSC_LIB are set. If they are not, the configuration process will be interrupted (otherwise you have to wait to the end of the compilation to find out that your program won't link).

Second section specifies the standard PETSc libs. eg.:

```
PETSC_LIB_DIR = $(PETSC_DIR)/lib/libg/$(PETSC_ARCH)
PETSC_LIBS    = petscts petscsnes petscsles petscdm
```

Third section adds platform dependent file, by checking PETSC_ARCH and assigning the appropriate libs.

In the end the variables are assigned to the variables that Cactus make process is using (note the incremental assignment "+=")

```
LIBDIRS    += $(PETSC_LIB_DIR) $(X_LIB_DIR)
LIBS       += $(PETSC_LIBS) $(PLATFORM_LIBS) X11
EXTRAFLAGS += -I$(PETSC_DIR)/include
```