About the inner workings of SGRID

- SGRID is written in pure C
- it is OpenMP parallelized
- SGRID is modular, it consists of
  - a main part that contains: coordinate transformations, functions to take spectral derivatives, Newton-Raphson solvers, etc.
  - one module for every physics problem such as binary neutron star initial data
- both main part and each module are controlled by parameters
- some parameters are simply physical parameters such as star masses
- other parameters control the choice of the number of grid points and numerical methods

Poisson3 - An example Module for solving two coupled non-linear elliptic equations

- solves 2 simple coupled Poisson like equations for ψ and χ:

$$\begin{aligned} \chi\partial_x^2\psi + \partial_y^2\psi + \chi^2\partial_z^2\psi &= \rho_1 \\ \partial_x^2\chi + (\partial_x\psi)\partial_y^2\chi + \partial_z^2\chi &= \rho_2 \end{aligned}$$

- contains 2 main C-files
  - `sgrid_Poisson3.c` :
    - determines which functions are called at what time
    - defines variables and parameters
  - `Poisson3.c` :
    - contains functions needed to solve the problem, e.g.:
      - `F_Poisson3` implements the 2 equations
      - `J_Poisson3` implements the linearized equations

$$\begin{aligned} \chi\partial_x^2\delta\psi + \delta\chi\partial_x^2\psi + \partial_y^2\delta\psi + \chi^2\partial_z^2\delta\psi + 2\delta\chi\partial_z^2\psi &= 0 \\ \partial_x^2\delta\chi + (\partial_x\psi)\partial_y^2\delta\chi + (\partial_x\delta\psi)\partial_y^2\chi + \partial_z^2\delta\chi &= 0 \end{aligned}$$

sgrid_Poisson3.c

```
int sgrid_Poisson3()
{
  /* functions */
  AddFun(PRE_COORDINATES, Poisson3_initboxes, "initialize boxes we use");
  AddFun(PRE_INITIALDATA, Poisson3_startup, "initialize Poisson3");
  AddFun(INITIALDATA, Poisson3_solve, "solve Poisson3 Eqs.");

  /* variables */
  AddVar("Poisson3_Psi",     "", "field that satisfies Eq 1: "
         "Chi Psixx + Psiyy + Chi Chi Psizz = rh1");
  AddVar("Poisson3_Psi",     "i",    "1st deriv of Psi");
  AddVar("Poisson3_Psi",     "(ij)", "2nd deriv of Psi");
  ...

  /* parameters */
  AddPar("Poisson3_itmax", "10", "maximal number of Newton iterations");
  AddPar("Poisson3_tol",   "1e-6","tolerance for Newton step or multigrid");
  AddPar("Poisson3_linSolver", "bicgstab", "linear solver used "
         "[bicgstab,UMFPACK,templates_GMRES_with_Jacobi_precon]");
  AddPar("Poisson3_linSolver_itmax", "20", "max num of linSolver iterations");
  AddPar("Poisson3_linSolver_tolFac","0.1", "tol factor for linSolver");
  AddPar("Poisson3_linSolver_Precon", "I",
         "Preconditioner used [I,fd_UMFPACK,templates]");
  AddPar("Poisson3_grid", "", "what grid we use [2starcubes,CubedSpheres]");
  ...
  return 0;
}
```

- each parameter has a default value that can be overridden in the parameter file

Poisson3.c

```
void F_Poisson3(tVarList *VLFu, tVarList *VLu, tVarList *VLuAll, tVarList *VLluAll)
{
  ...
  forallboxes(grid, b) {
    ...
    /* compute the derivs */
    D_and_DD_of_S(box, VLu->index[0], VLuAll->index[1], VLuAll->index[4]);
    D_and_DD_of_S(box, VLu->index[1], VLuAll->index[11], VLuAll->index[14]);

    /* Poisson3 eqs */
    forallpoints(box, i) {
      FPsi[i] = Chi[i]*Psixx[i] + Psiyy[i] + Chi[i]*Chi[i]*Psizz[i] - rh1[i];
      FChi[i] = Chixx[i] + Psix[i]*Chiyy[i] + Chizz[i] - rh2[i];
    }
  }
  ...
}
void J_Poisson3(tVarList *VLJlu, tVarList *VLlu, tVarList *VLuAll, tVarList *VLluAll)
{
  ...
    /* linearized Poisson3 eqs */
    forallpoints(box, i) {
      JlPsi[i] = Chi[i]*lPsixx[i] + lChi[i]*Psixx[i]  +  lPsiyy[i]  +
                 Chi[i]*Chi[i]*lPsizz[i] + 2.0*lChi[i]*Psizz[i];
      JlChi[i] = lChixx[i] + Psix[i]*lChiyy[i] + lPsix[i]*Chiyy[i] + lChizz[i];
    }
  ...
}
```

- can switch on OpenMP parallelization for any of the loops shown here

An example parameter file for Poisson3

```
physics = Poisson3

Poisson3_linSolver_itmax  = 100000
Poisson3_linSolver        = templates_GMRES_with_BlockJacobi_precon
Poisson3_linSolver_Precon = templates
Poisson3_linSolver_tolFac = 0.001

nboxes = 2

box0_min1 = 0
box0_max1 = 1

box0_min2 = -1
box0_max2 = +1
...

n1 = 8
n2 = 8
n3 = 8

1douttime   = 0.1
1doutput    = Poisson3_Psi Poisson3_Psix Poisson3_Chi Poisson3_Chix
1doutputall = yes
```