Pgen files—other functions

List of pgen files with other functions besides ProblemGenerator and some Initializer functions. Many of these pgen files had initializer functions that may have been excluded if the function did not show any major differences besides declaring variables/constants. Purpose is to seek out functions that may have a time component being either recorded or printed onto the screen corresponding to some calculated value.

Beam cpp

- TwoBeams sets coordinates, Athena array, time as variable, along with radiation and other integers for is, ie, etc
  - Calculates slope for different angles and frequencies
- TwoBeamHydro again similar to TwoBeams except with hydro elements
- Mesh::InitUserMeshData based on user refinement condition and is used based on nr radiation or im radiation being enabled
- MeshBlock::InitUserMeshBlockData specifies an angle and octnum pointer
- It does not appear that any of these use time specified arguments with regard to actually printing out variables with time

binary_gravity

- Mesh::UserWorkAfterLoop
  - Defines x1, r, m1, G, and densities, as well as i, j, and ks coordinates
  - Adds cycle argument that tells how many times solver has been activated
  - Also includes a start time and clock variable
  - Lots of conditional ifdef or fiend statements that are dependent on poisson solvers
  - Much of the print statements t terminals invoice the time it took to calculate the poisson solver, along with the number of zones in mesh, mesh configuration, etc

Blast

- Mesh::UserWorkAfterLoop
  - Sets x1, x2, and x3 coordinates, as well as if statements for different coordinate systems
  - Sets rmax, rmin, and rave initial 0 value
  - Describes physical grid spacing for blast center
- RefinementCondition
  - Describes the maximum pressure gradient

chem_H2

- HistoryT outputs average temperature
- Mesh::UserWorkAfterLoo
  - Reads off and assigns various parameters, including heating and cooling

chem_onecell_sixray

- Inner/outer boundary descriptions for x1, x2, and x3
- Separate function for each of these after the problem generator function

Collapse

- SOurceMask assigns coordinate/mesh values to x, y, and z and calculates r^2
- JeansCondition calculates refinement condition but does not include anything time dependent
- BEProfile
- Cooling sets boundary conditions

cr_diffusion

- Mesh::UserWorkAfterLoop
  - Time is found going through the entire mesh block and calculate the error of the solution
- Diffusion
  - Calculates diffusion in different components

Disk

- Six disk inner and outer functions to define coordinate mesh components and time components
- Mesh::InitUserMeshData
  - Initializes data and boundary conditions
- GetCylCoord = get cylindrical coordinates
- DenProfileCyl = density profile in cylindrical coordinate
- PoverR = pressure/density computation
- VelProfileCyl = rotational velocity
- Goes back to the six disk inner and outer functions to obtain solution in ghost zones to initialize values

DMR

- Three inner/outer conditions for x1 and x3
- RefinementCondition = maximum density and pressure curvature

fft

- Mesh::UserWorkAfterLoop
  - Calculates start and end times/duration of each run I think (cpu time)

field_loop_poles

- VelProfileCyl = calculates velocity of 3 components
- Loop[Inner/Outer][X#] = defining boundary conditions of different components

gr_mhd_inflow

- CalculateFromTable = with set of variables, interpolate location, calculate velocity, and calculate covariant magnetic field from table

gr_linear_wave

- Mesh::UserWorkAfterLoop = calculates errors of all variables calculated and write it to a file

gr_torus

- MeshBlock::UserWorkInLoop = storing history quantities for output
- MeshBlock::UserWorkBeforeOutput = storing cell quantities for output
- InflowBoundary = describes inflow boundary conditions
- OutflowBoundary = describes outflow boundary conditions
- Other functions include calculating various variables, none of which include time as an output

gr_bondi

- Most of these are to compute either boundary conditions or, temperature, radio, etc

jgg

- Mesh::UserWorkInLoop = checks output per unit of time

```
void Mesh::UserWorkInLoop() {
  bool flag = false;
  // check output
  Real present_time = time + dt;
  if (error_output) {
    // check flag
    if ((present_time < tlim) && (nlim < 0 || (ncycle + 1) < nlim)
        && (present_time > hst_next_time)) {
      flag = true;
      hst_next_time += hst_dt;
    }
    if ((present_time >= tlim) || (nlim >= 0 && (ncycle + 1) >= nlim)) {
      flag = true;
    }
  }
```

  - Computes mesh components

Jet

- JetInnerX1= calculates jet for inner x1 boundary

Poisson

- Mesh::UserWorkAfterLoop = prints out cycle numbers as it calculates mesh components

```
  if (Globals::my_rank == 0) {
    std::cout << "Timing Possison Solver                                  " <<
        std::endl;
    std::cout << "number of zones in Mesh = " << zones << std::endl;
    std::cout << "Mesh configuration = " << mesh_size.nx1 << "x"
              << mesh_size.nx2 << "x" << mesh_size.nx3 << std::endl;
    std::cout << "number of zones in MeshBlock = " << mb_zones << std::endl;
    std::cout << "MeshBlock configuration = " << pmb->block_size.nx1 << "x"
              << pmb->block_size.nx2 << "x" << pmb->block_size.nx3 << std::endl;
    std::cout << "number of processors = " << Globals::nranks << std::endl;
    std::cout << "processor configuration = "
              << nrbx1 << "x" << nrbx2 << "x" << nrbx3 << std::endl;
    std::cout << "cpu time used  = " << cpu_time << std::endl;
    std::cout << "cpu time used/cycle  = " << cpu_time/static_cast<Real>(ncycle)
              << std::endl;
    std::cout << "zone-cycles/cpu_second = " << zc_cpus << std::endl;
    std::cout << "zone-cycles(NlogN)/cpu_second = " << zc_cpus2 << std::endl;
lef OPENMP_PARALLEL
    std::cout << "=========================================================" <<
        std::endl;
    float zc_omps = static_cast<Real>(zones*ncycle)/omp_time;
    std::cout << "omp number of threads = " << GetNumMeshThreads() << std::endl;
    std::cout << "omp wtime used = " << omp_time << std::endl;
    std::cout << "zone-cycles/omp_wsecond = " << zc_omps << std::endl;
lif
    std::cout << "=========================================================" <<
        std::endl;
  }

  Real err1 = 0.0, maxphi = 0.0; // err2 = 0.0
```

rad_linearwave

- Mesh::UserWorkAfterLoop
  - Calculate errors and write it to a file
- Mesh::InitUserMeshData
  - Calculate refinement condition for density curvature

Rt

- Pressure for inner and outer x mesh coordinates
- Mesh::InitUserMeshData = initialize boundary conditions and pressures
- ProjectPressure[inner/outer][X#] = pressure integrated into various cells

shk_cloud

- ShockCloudInnerX1 = fixes BCson L-x1 of grid to post shock flow

slotted_cylinder

- Mesh::InitUserMeshData = sets radius and mesh coordinates
- Mesh::UserWorkAfterLoop = calculate errors

ssheet

- Mesh::InitUserMeshData = reads in mesh coordinates/initializes coordinates from input files
- Mesh::UserWorkInLoop = calculates error output based on time,

initializes velocities

Strat

- Many functions defining vertical gravity and Strat/history of flow through mesh coordinates that have unused arguments
- MeshBlck::USerWorkInLoop = sets various variables
- None of these include time

thermal_multigroup

- Mesh::UserWorkAfterLoop
  - Calculate the sum of radiation energy density across the mesh for each frequency
  - Gives error warnings