SAMRAI Input Parameters and Restart

Introduction

Many of the classes in the SAMRAI library are controlled via input parameters specified in an input database. Some of these classes require input from the user. Other classes completely define a default behavior and do not require user input but the user may override this default behavior by specifying input parameters in the input database.

Currently, all of SAMRAI's classes that accept input parameters do not write their state to the restart database. This requires that when running a problem from a restart database an input database is also required so that these classes can be configured as the user intends.

In the future it is likely that SAMRAI will move toward a more consistent model in which every library class accepting input parameters writes its state to the restart database. In the mean time, it is necessary to provide some guidance to users concerning running from a restart database especially when using an input database that has changed since the restart database was created.

Classes Which Do Not Write To Restart

The following classes accept input parameters but do not write any state to restart:

- solv::CellPoissonFACSolver
- solv::FACPrecoditoner
- solv::CellPossonFACOps
- solv::CellPoissonHypreSolver
- solv::LocationIndexRobinBcCoefs
- tbox::TimerManager
- mesh::TreeLoadBalancer
- mesh::ChopAndPackLoadBalancer
- mesh::StandardTagAndInitialize
- mesh::BergerRigoutsos
- hier::PersistentOverlapConnectors
- appu::CartesianBoundaryUtilities2
- appu::CartesianBoundaryUtilities3

Since no state is saved to restart the user is free to change any input parameter for theses classes in their input database. For *most* of these classes this will not cause

any problems. There is no inherent incompatibility in solving, clustering, or load balancing differently going forward from a restart database. However, changing boundary conditions is likely **not** a good idea. Therefore it is strongly advised that input parameters for CartesianBoundaryUtilities[2,3] **not** be modified when restarting a problem. Since these classes do not save their state to restart the library cannot currently ensure that users are not using different inputs for these classes.

Classes That Do Write To Restart

The following classes accept input parameters and do write their state to restart:

- solv::SNES_SAMRAIContext
- solv::KINSOL_SAMRAIContext
- mesh::GriddingAlgorithm
- geom::GridGeometry
- geom::CartesianGridGeometry
- hier::PatchHierarchy
- algs::ImplicitIntegrator
- algs::TimeRefinementIntegrator
- algs::HyperbolicLevelIntegrator
- algs::MethodOfLinesIntegrator

Again, users may wish to change input parameters for these classes on restart. They may also be perfectly content to run the problem with no changes to these classes. To indicate that a class should read its input database for new input parameters set "read_on_restart" to TRUE in that class' input database. The default for this parameter is FALSE so to run with no changes no action is needed.

There are restrictions on which parameters may be modified on restart and how they may be modified. For example, it is not allowed to change anything about GridGeometry or CartesianGridGeometry as to do so would change the problem in such a fundamental way that it would not be possible to restart. At the other extreme, any input parameter for SNES_SAMRAIContext or KINSOL_SAMRAIContext may be overridden on restart. In any event, since these classes write their state to restart, the library can enforce each class' rules for what may be modified on restart and how it may be modified.

In all cases, when these classes are constructed they first read their input parameters from the restart database. Then they look at the input database to see if read_on_restart is TRUE. If it is, the class reads the input parameters in the input database and performs the necessary checks for valid input and valid input changes. As when a problem is not run from an input database, invalid input will result in an unrecoverable error. Invalid changes to input parameters will result in a warning

being issued. The code will ignore the invalid input and continue using the input parameter from the restart database.

Input Parameter Documentation

Each class documents its input parameters. This is in the form of doxygen comments in the class header file. Refer to the library source code documentation that forms the "SAMRAI-docs" tarball for this information. In addition to describing what each parameter controls and important caveats and interactions each class contains a table describing each parameter's type, default, range, whether it is optional or required, and its behavior on restart.