Poseidon Manual

Nick Roberts

January 10, 2024
Poseidon v0.4
Manual v0.1

Contents

Chapter 1

Introduction to Poseidon

Poseidon is a open-source gravity solver designed for use with core-collapse supernova and other astrophysical simulations.

Chapter 2

Getting Started

2.1 Downloading Poseidon

The source code is available at https://github.com/jrober50/Poseidon.

2.2 Running the Yahil Collapse Problem

Poseidon has been designed to be used by other programs and not as a stand alone code. Therefore, basic drivers that create test sources and call Poseidon have been provided in `Poseidon/Drivers` (NR: Finish)

2.3 Building with Poseidon

2.3.1 Codes using Fortran Arrays

In the calling program's makefile, add `include $(POSEIDON_DIR)/Build/Make.Poseidon_Native`. This gives the makefile system access to the lists of Poseidon files and includes the necessary directories. `POSEIDON_o` is the list of objects files that are generally needed for compilation. While `POSEIDON` is the list of Poseidon files with their regular file suffixes. (NR: There are also lists that just contain the f90, F90, and cpp files, but I want to rebrand them. But I want to double check if that'd break something first.)

2.3.2 Codes using `AMReX`

Programs that use `AMReX` are built using GNUmake. To include Poseidon, add '`include $(POSEIDON_DIR)/Build/Make.Poseidon`' in the GNUmake file.

Chapter 3

Basics of Interfacing With Poseidon - Newtonian Solver

Performing a Newtonian solve with Poseidon involves 6 steps, shown in figure 3.1.

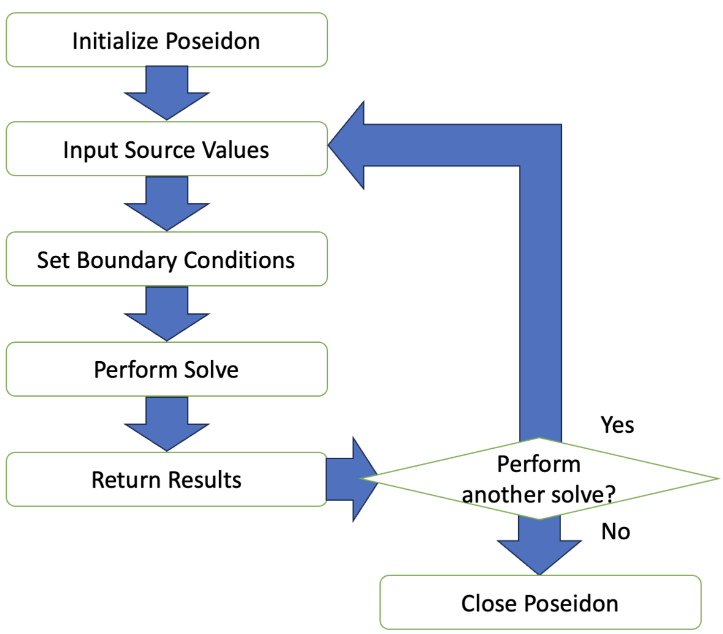

Figure 3.1: Order of operations for a Newtonian solve with Poseidon.

Each step will be detailed in the following sections.

3.1 Initialize Poseidon

The first step to interfacing with Poseidon is calling `Initialize_Poseidon` which is located in `Module Poseidon_Interface_Initialization`.

```
CALL Initialize_Poseidon &
        (    Source_NQ,                          &
             Source_xL,                          &
             Source_RQ_xlocs,                    &
             Source_TQ_xlocs,                    &
             Source_PQ_xlocs,                    &
             Source_Units,                       &
             Source_Radial_Boundary_Units,       &
             Newtonian_Mode_Option = .TRUE.,     &
             ...
        )
```

This routine requires the user to define some quantities that Poseidon needs to create it's data space. Further inputs can be submitted to further specify how Poseidon will operate. In this section, we will discuss the required inputs and high light a few of the more important optional inputs. A full list of the arguments for `Initialize_Poseidon` can be found in 5.1.

3.1.1 Required Arguments

There are seven inputs required when calling `Initialize_Poseidon`:

- `Source_NQ` is a rank 1 integer array of length 3. Each entry defines the number of source points per element in each dimensions.
- `Source_xL` is a rank 1 array of reals of length 2. The first entry defines the calling code's lower bound of it's integration space, while the second defines the upper bound.
- `Source_RQ_xlocs` is a rank 1 real array of length Source_NQ(1). This array contains the locations of source data radial nodes in the calling program's integration space.
- `Source_TQ_xlocs` is a rank 1 real array of length Source_NQ(2). This array contains the locations of source data theta nodes in the calling program's integration space.
- `Source_PQ_xlocs` is a rank 1 real array of length Source_NQ(3). This array contains the locations of source data phi nodes in the calling program's integration space.

- `Source_Units` is a character of length 1. This input accepts four values, each for a different common set of units. The accepted inputs are "C" for CGS units, "S" for MKS units, "G" for geometrized units, and "U" for unit-less.
- `Source_Radial_Boundary_Units` is a character of length 2. This input accepts three values, each for a different radial distance unit. The accepted inputs are "cm", " m", and "km".
- `Newtonian_Mode_Option` is a logical value that must be set to `.TRUE.`. By default, Newontian_Mode is `.FALSE.` and Poseidon will attempt a xCFC solve. Therefore it is essential to include this when performing a Newtonian solve with Poseidon.

3.1.2 Optional Arguments

Domain Decomposition Specifics

Poseidon is designed to interface with codes that use AMReX mesh refinement and datatypes as well as native Fortran datatypes. When the calling program uses AMReX, details about the domain composition are contained in the source data multifabs and do not need to be set during initialization. If the calling program does not use AMReX, then there are a few optional inputs to `Initialize_Poseidon` that need to be set to accurately represent a problem within Poseidon.

These inputs are:

- Source_NE_Option is a rank 1 integer array of length 3. Each entry specifies the number of elements in a radial, theta, and phi dimensions accordingly.
- Domain_Edge_Option is a rank 1 double precision real array of length 2. The entries specify the inner and outer radii of the domain.
- Source_R_Option is a rank 1 double precision real array of length Source_NE_Option(1)+1. The values of this array are equal to the edge values of each radial element.
- Source_T_Option is a rank 1 double precision real array of length Source_NE_Option(2)+1. The values of this array are equal to the edge values of each theta element.
- Source_P_Option is a rank 1 double precision real array of length Source_NE_Option(3)+1. The values of this array are equal to the edge values of each phi element.
- Source_DR_Option is a rank 1 double precision real array of length Source_NE_Option(1). The values of this array are equal to the widths of each radial element.

- Source_DT_Option is a rank 1 double precision real array of length Source_NE_Option(2). The values of this array are equal to the widths of each radial element.
- Source_DP_Option is a rank 1 double precision real array of length Source_NE_Option(3). The values of this array are equal to the widths of each radial element.

Poseidon only requires that either the edge values or widths of the elements are set, not both. The user can provide whichever is most convenient.

Solver Parameters

Poseidon uses several numerical methods during each solve. Some of these methods have limits or parameters that effect how they perform. These parameters can be set when calling `Initialize_Poseidon` using the following arguments.

- FEM_Degree_Option is a scalar integer input that sets the degree of the finite element expansion.
- L_Limit_Option is a scalar integer input that sets the degree of the spherical harmonic expansion.

For codes that use AMReX, each of the above arguments can be set in the inputs file using the `poseidon.` specifier.

3.2 Input Source Values

For Poseidon to perform a Newtonian solve, it must be provided with the matter density of problem. (NR: Include eq, specify want rho?) To submit this information to Poseidon, the `Poseidon_Input_Sources` routine is used. This routine has been overloaded to accept both AMReX multifabs as well as native Fortran arrays.

If the calling program is using AMReX datatypes, then `Poseidon_Input_Sources` takes the form,

```
CALL Poseidon_Input_Sources(   MF_Source  )
```

where MF_Source is the source multifab. (NR: Discuss packing of multifab?)

Likewise, if the calling program uses native Fortran arrays, the `Poseidon_Input_Sources` takes the form,

```
CALL Poseidon_Input_Sources(   Input_Rho  )
```

(NR: Rebrand to Input_Source or Source_Input) Input_Rho is a rank 4 double precision real array. The dimensionality of Input_Rho is $(N_{DOF}, Source\_NE(1), Source\_NE(2), Source\_NE(3))$, where $N_{DOF} = Source\_NQ(1) * Source\_NQ(2) * Source\_NQ(3)$.

3.3 Set Boundary Conditions

For Poseidon to function, it needs boundary values to impose on the problem to make the solution unique. Currently, Poseidon is limited to uniform boundary conditions, which are set using the `Poseidon_Set_Uniform_Boundary_Conditions` routine.

```
CALL Poseidon_Set_Uniform_Boundary_Conditions &
(    BC_Location_Input ,  &
     BC_Type_Input ,      &
     BC_Value_Input       &
 )
```

All three inputs are required and are defined as

- BC_Location_Input is a length 1 character. Either "I" or "O" for inner or outer boundary respectively. This tells Poseidon to which boundary the following conditions will be applied.
- BC_Type_Input is a length 1 character. Either "N" or "D" for Neumann or Dirichlet respectively. This tells Poseidon what kind of boundary condition is being set.
- BC_Value_Input is a rank 1 array of length 1 for a Newtonian solve. Each entry should contain the value to be enforced at the specified boundary. (Note: This routine is overloaded to also accept an array of length 5 for xCFC type solves.)

Poseidon views the domain as having two boundaries, an inner boundary and an outer boundary. As Poseidon works in a spherical polar domain, the inner boundary corresponds to the points at smallest radial distance from the origin in the domain. Likewise the outer boundary corresponds to all points at the largest radial point contain in the domain. For each of these boundaries, the routine `Poseidon_Set_Uniform_Boundary_Conditions` needs to be called to set the value for that domain. Therefore `Poseidon_Set_Uniform_Boundary_Conditions` must be called twice, once for the inner boundary and once for the outer boundary.

3.4 Perform Solve

Once the previous steps have been performed, Poseidon is ready to perform the solve using the `Poseidon_Run` routine.

```
CALL Poseidon_Run( )
```

This routine is found in `Module Poseidon_Interface_Run` and has no arguments. When called, this routine will attempt to solve the Newtonian gravitational equation using the sources provided and subject to the defined boundary conditions. When completed, the solution coefficients will be known and the solution can be requested.

3.5 Return Results

As in Section 3.2, the routine for returning results from Poseidon, `Poseidon_Return_Newtonian_Potential` has been overloaded to return results in both AMReX multifabs and native Fortran arrays.

If the calling program is using AMReX datatypes, then `Poseidon_Return_Newtonian_Potential` takes the form,

```
CALL Poseidon_Return_Newtonian_Potential( MF_Results )
```

where MF_Results is a multifab with the same domain decomposition as the multifab used to submit the source, MF_Sourece. (NR: Discuss packing of multifab?)

Likewise, if the calling program uses native Fortran arrays, `Poseidon_Input_Sources` takes the form,

```
CALL Poseidon_Input_Sources( Return_Potential )
```

Return_Potential is a rank 4 double precision real array. The dimensionality of Return_Potential is $(N_{DOF}, Source\_NE(1), Source\_NE(2), Source\_NE(3))$, where $N_{DOF} = Source\_NQ(1) * Source\_NQ(2) * Source\_NQ(3)$.

3.6 Close Poseidon

When Poseidon is no longer needed, it should be properly closed. This is accomplished using the `Poseidon_Close` routine.

```
CALL Poseidon_Close( )
```

`Poseidon_Close` is found in `Module Poseidon_Interface_Close` and has no arguments. This routine deallocates Poseidon data structures therefore all information about the solution will be lost once this is called.

Chapter 4

Basics of Interfacing With Poseidon - xCFC Solver

Performing a xCFC solve with Poseidon involves steps, shown in figure 4.1.
Each step will be detailed in the following sections.

4.1 Initialize Poseidon

The first step to interfacing with Poseidon is calling `Initialize_Poseidon` which is located in `Module Poseidon_Interface_Initialization`.

```
CALL Initialize_Poseidon &
        (       Source_NQ,                       &
                Source_xL,                       &
                Source_RQ_xlocs,                 &
                Source_TQ_xlocs,                 &
                Source_PQ_xlocs,                 &
                Source_Units,                    &
                Source_Radial_Boundary_Units,    &
                ...
        )
```

This routine requires the user to define some quantities that Poseidon needs to create it's data space. Further inputs can be submitted to further specify how Poseidon will operate. In this section, we will discuss the required inputs and high light a few of the more important optional inputs. A full list of the arguments for `Initialize_Poseidon` can be found in 5.1.

4.1.1 Required Arguments

There are seven inputs required when calling `Initialize_Poseidon`:

- `Source_NQ` is a rank 1 integer array of length 3. Each entry defines the number of source points per element in each dimensions.
- `Source_xL` is a rank 1 array of reals of length 2. The first entry defines the calling code's lower bound of it's integration space, while the second defines the upper bound.
- `Source_RQ_xlocs` is a rank 1 real array of length Source_NQ(1). This array contains the locations of source data radial nodes in the calling program's integration space.
- `Source_TQ_xlocs` is a rank 1 real array of length Source_NQ(2). This array contains the locations of source data theta nodes in the calling program's integration space.
- `Source_PQ_xlocs` is a rank 1 real array of length Source_NQ(3). This array contains the locations of source data phi nodes in the calling program's integration space.
- `Source_Units` is a character of length 1. This input accepts four values, each for a different common set of units. The accepted inputs are "C" for CGS units, "S" for MKS units, "G" for geometrized units, and "U" for unit-less.
- `Source_Radial_Boundary_Units` is a character of length 2. This input accepts three values, each for a different radial distance unit. The accepted inputs are "cm", " m", and "km".

4.1.2 Optional Arguments

Domain Decomposition Specifics

Poseidon is designed to interface with codes that use AMReX mesh refinement and datatypes as well as native Fortran datatypes. When the calling program uses AMReX, details about the domain composition are contained in the source data multifabs and do not need to be set during initialization. If the calling program does not use AMReX, then there are a few optional inputs to `Initialize_Poseidon` that need to be set to accurately represent a problem within Poseidon.

These inputs are:

- Source_NE_Option is a rank 1 integer array of length 3. Each entry specifies the number of elements in a radial, theta, and phi dimensions accordingly.
- Domain_Edge_Option is a rank 1 double precision real array of length 2. The entries specify the inner and outer radii of the domain.

- Source_R_Option is a rank 1 double precision real array of length Source_NE_Option(1)+1. The values of this array are equal to the edge values of each radial element.
- Source_T_Option is a rank 1 double precision real array of length Source_NE_Option(2)+1. The values of this array are equal to the edge values of each theta element.
- Source_P_Option is a rank 1 double precision real array of length Source_NE_Option(3)+1. The values of this array are equal to the edge values of each phi element.
- Source_DR_Option is a rank 1 double precision real array of length Source_NE_Option(1). The values of this array are equal to the widths of each radial element.
- Source_DT_Option is a rank 1 double precision real array of length Source_NE_Option(2). The values of this array are equal to the widths of each radial element.
- Source_DP_Option is a rank 1 double precision real array of length Source_NE_Option(3). The values of this array are equal to the widths of each radial element.

Poseidon only requires that either the edge values or widths of the elements are set, not both. The user can provide whichever is most convenient.

Solver Parameters

Poseidon uses several numerical methods during each solve. Some of these methods have limits or parameters that effect how they perform. These parameters can be set when calling `Initialize_Poseidon` using the following arguments.

- FEM_Degree_Option is a scalar integer input that sets the degree of the finite element expansion.
- L_Limit_Option is a scalar integer input that sets the degree of the spherical harmonic expansion.

For codes that use AMReX, each of the above arguments can be set in the inputs file using the `poseidon.` specifier.

4.2 Input Source Values Part I

4.3 Set Boundary Conditions

For Poseidon to function, it needs boundary values to impose on the problem to make the solution unique. Currently, Poseidon is limited to uniform boundary conditions, which are set using the `Poseidon_Set_Uniform_Boundary_Conditions` routine.

```
CALL Poseidon_Set_Uniform_Boundary_Conditions &
(     BC_Location_Input , &
      BC_Type_Input ,     &
      BC_Value_Input      &
 )
```

All three inputs are required and are defined as

- BC_Location_Input is a length 1 character. Either "I" or "O" for inner or outer boundary respectively. This tells Poseidon to which boundary the following conditions will be applied.
- BC_Type_Input is a length 1 character. Either "N" or "D" for Neumann or Dirichlet respectively. This tells Poseidon what kind of boundary condition is being set.
- BC_Value_Input is a rank 1 array of length 5 for a Newtonian solve. Each entry should contain the value to be enforced at the specified boundary. (Note: This routine is overloaded to also accept an array of length 1 for Newtonian solves.)

Poseidon views the domain as having two boundaries, an inner boundary and an outer boundary. As Poseidon works in a spherical polar domain, the inner boundary corresponds to the points at smallest radial distance from the origin in the domain. Likewise the outer boundary corresponds to all points at the largest radial point contain in the domain. For each of these boundaries, the routine `Poseidon_Set_Uniform_Boundary_Conditions` needs to be called to set the value for that domain. Therefore `Poseidon_Set_Uniform_Boundary_Conditions` must be called twice, once for the inner boundary and once for the outer boundary.

4.4 Set Initial Guess

4.5 Solve for and Return the Conformal Factor

4.6 Calculate Remaining Source Values

4.7 Input Source Values Part II

4.8 Perform Solve

Once the previous steps have been performed, Poseidon is ready to perform the solve using the `Poseidon_Run` routine.

```
CALL Poseidon_Run( )
```

This routine is found in `Module Poseidon_Interface_Run` and has no arguments. When called, this routine will attempt to solve the xCFC system of equations using the sources provided and subject to the defined boundary conditions. When completed, the solution coefficients will be known and the solution can be requested.

4.9 Return Results

4.10 Close Poseidon

When Poseidon is no longer needed, it should be properly closed. This is accomplished using the `Poseidon_Close` routine.

```
CALL Poseidon_Close( )
```

`Poseidon_Close` is found in `Module Poseidon_Interface_Close` and has no arguments. This routine deallocates Poseidon data structures therefore all information about the solution will be lost once this is called.

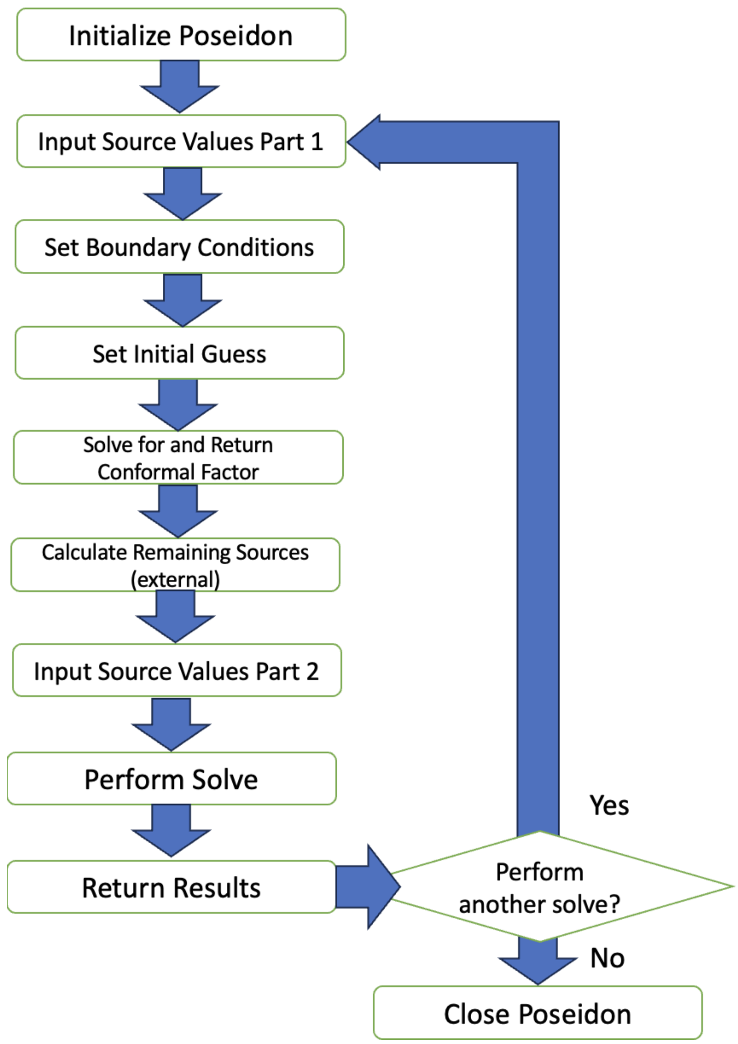

Figure 4.1: Order of operations for a xCFC solve with Poseidon.

Chapter 5

Top Level Routines

These are the top level routines that most users will need to interface with Poseidon.

5.1 Initialize_Poseidon

5.2 Poseidon_Input_Sources

5.3 Poseidon_Initialize_Flat_Guess

5.4 Poseidon_Set_Uniform_Boundary_Conditions

5.5 Poseidon_Return_*