ASYNCH Documentation

Scott J. Small

September 1, 2014

!!!! Need Link description !!!! !!!! Give ODE description !!!! !!!! Walk through an example or two with setting up a model !!!!

1 Global File Structure

!!!! For .dbc options, explain what is needed for the queries. !!!!

Global files (.gbl) are used to specify all inputs for the solvers. This includes river network topology files, database connections, initial model states, what information is printed to output files, model forcings, etc. Global files have a very rigid structure, unlike input XML files, and information must be specified in a particular order. The description of each input of the global files below are given in the order in which they are to be specified in the actual file.

Global files are always ASCII files, assumed to be in UNIX format. The percent sign (%) is used for single line comments. As decribed below, certain inputs are expected to be given within a single line. Other than this restriction, white space is ignored. Arguments surrounded by { } below are mandatory, while those surrounded by the square brackets [ ] are dependent upon values specified by other parameters.

1.1 Model Type and Maxtime

Format:
{model id} {total simulation time}

The first value specifies the id for the model to be used. This is a non-negative integer value which corresponds to a particular system of ordinary-differential equations (or possibly DAEs). A list of pre-existing models is given in !!!! blah !!!!.

The second is the total simulated time used by the solvers. This is a floating point number. The units are generally in minutes, however, some models can get away with using other units (seconds, hours, etc) if carefully crafted. All built in models use minutes for the time.

1.2 Parameters on Filenames

Format:
{parameter on output filename flag}

This is a boolean value (0 or 1) that indicates whether all output filenames should include the uniform in space and time parameters. 0 indicates no, 1 indicates yes. This feature can be useful for keeping track of output files.

1.3 Solver Outputs

Format:
{number of outputs}
[output1]
[output2]
[...]

This set of input parameters specifies the names of all outputs from the solvers. Several built in outputs exist, and the user is able to construct his own outputs. Built in outputs are given in table !!!! blah !!!!. Output names are case sensitive. The first required value is the number of outputs ($>= 0$), followed by the names of each output, on separate lines.

1.4 Peakflow Statistics Function Name

Format:
{function name}

This sets the function for outputting peakflow information. The built in peakflow function “Classic” is one option, and the user is free to construct his own. A function name must be specified here, even if peakflow information is not requested for any hillslopes.

1.5 Global Parameters

Format:
{number of parameters} [parameter1] [parameter2] ...

This is where model parameters which are constant in space and time are specified. The first value is a nonnegative integer specifying the number of global parameters to follow. Every model requires a certain number of global parameters. If the number given in the global file is less than expected for a particular model, an error occurs. If the number is greater than expected, a warning is given. These “extra” parameters are available to the model for use. This can sometimes be useful for quick tests, but should be avoided normally.

The parameter meanings depend upon the model used. The units of these parameters is also model dependent.

1.6 Buffer Sizes

*F*ormat:
{steps stored at each link} {max number of steps transferred} {discontinuity buffer size}

These nonnegative integer values allow the user to specify sizes of internal buffers. In general, as these numbers are increased, the solvers run faster, but more memory is required. A good starting point that works in most cases is the set 30 10 30. Typically, if these values need to be less than 10 to run the solvers, a deeper issue with memory constraints should be addressed.

1.7 Topology

*F*ormat:
{topology flag} [output link id] {.rvr filename or .dbc filename}

This is where connectivity of the river network is specified. This can be done in one of two ways. If the topology flag is 0, a river topology file (.rvr) is used. If the topology flag is 1, then topology is downloaded from the database specified with the .dbc filename. The database connection allows for one additional feature: a subbasin can be specified. If the output link id is taken to be 0, all link ids found in the database are used. Otherwise, the hillslope with link id specified and all upstream hillslopes are used. Pulling subbasins from a topology file is not currently supported.

1.8 Hillslope Parameters

*F*ormat:
{parameter flag} {.prm filename or .dbc filename}

This specifies where parameters which vary by location, but not time, are specified. If the parameter flag is 0, the paramters are given in a parameter (.prm) file. If the flag is 1, then the parameters are downloaded from the database specified by the database connection file (.dbc). The number, order, meaning, and units of these parameters varies from model to model.

1.9 Initial States

*F*ormat:
{initial state flag} {.ini, .uini, .rec, or .dbc filename} [unix time]

This section specifies the initial state of the model. The values for the initial state flag can be 0, 1, 2, or 3, corresponding, respectively, to a .ini, .uini, .rec, or .dbc file. The unix time argument is used for database connections only. This value is available in the query of the database connection file and can be used to for selecting values from tables.

1.10 Forcings

*F*ormat:

{number of forcings}
[forcing1 flag] [forcing1 information]
[forcing2 flag] [forcing2 information]
[...]

Information about time dependent forcings is specified here. Each model has an expected number of forcings. If the number of forcings specified here is less than expected, an error is thrown. If the number of forcings is greater than expected, a warning is given. This warning allows for tests to be performed and implemented quickly. In general, this feature should be avoided.

Forcing information varies considerably based upon the corresponding forcing flag. Several forcing types require unix times to determine what forcing data to use. If a model requires multiple forcings with unix times, the times *do not* need to be consistent, i.e., one forcing could start on July 1st 2014 at midnight, while another forcing starts at April 5th 2008.

1.10.1 No Forcing

Format:
0

A forcing flag of 0 specifies no forcing input. This is the same as a forcing value of 0.0 for all hillslopes and all time.

1.10.2 Storm File

Format:
1 {.str filename}

A forcing flag of 1 indicates the forcing is specified by a .str file. The filename and path of a valid storm (.str) file is required.

1.10.3 Binary Files

Format:
2 {binary file identifier}
{chunk size} {time resolution} {beginning file index} {ending file index}

A forcing flag of 2 indicates the forcing is specified by a collection of binary forcing files. The identifier can be adorned with a path to the binary files. The chunk size is a postive integer that indicates the number of binary files kept in memory at once. The time resolution indicates the amount of time between successively indexed binary files. This value is a floating point number with units equal to those of the time variable of the model used. The beginning and ending file indices indicate the range of the binary files to be used. The indices are integer valued.

The simulation will begin using the binary file with index given by the beginning file index. If the total simulation time would require binary files with index greater than the ending file index, the forcing values are taken to be 0.0 for all such binary files.

1.10.4 Forcings from Databases

*F*ormat:
3 {.dbc filename}
{chunk size} {time resolution} {beginning unix time} {ending unix time}

A forcing flag of 3 indicates the forcing data will be pulled from a PostgreSQL database. The database connection filename can include a path. The chunck size is a positive integer representing the number of forcing values pulled from the database at once from each hillslope. A chunck size of 10 tends to work well. A larger chunck size requires more memory and larger datasets returned from the database, but a small number of queries. The time resolution is a floating point number with units in minutes. This represents the time resolution of the data in the accessed table. The integrity of the database is not thoroughly checked by the solvers.

The simulation will begin using the data from the database with unix time given by the beginning unix time. If the total simulation time would require data from the database with unix time greater than the ending unix time, the forcing values are taken to be 0.0 for times greater than the ending unix time.

1.10.5 Uniform Forcings

*F*ormat:
4 {.ustr filename}

A forcing flag of 4 indicates a forcing that is uniform in space. The forcings are given by a uniform storm file (.ustr).

1.10.6 Forcings from Databases, with Forecasting

*F*ormat:
5 {.dbc filename}
{chunk size} {time resolution} {beginning unix time} {ending unix time}
{combined times}
{termination filename}

A forcing flag of 5 indicates forcing data will be pulled from a database, with forecasting mode. All required inputs are the same as forcing flag 3 with databases, but with a two extra paramters. The combined times parameter is a positive integer that represents the total number of timesteps used from the database in a single forecast. If there are less than this number of timestamps available, the solvers will wait until the database has enough. The termination file contains only a single number (0 or 1), and is created by the solvers. Initially, the file contains a 0. When this value is changed to a 1 by an external program, the solvers will halt making further forecasts, and begin termination routines. This is the proper way to stop the solvers.

In the future, this option will be removed.

1.10.7 GZipped Binary Files

*F*ormat:
6 {gzipped binary file identifier}
{chunk size} {time resolution} {beginning file index} {ending file index}

A forcing flag of 6 indicates the forcing is specified by a collection of binary forcing files that have been gzipped (compressed as .gz files). All parameters for this flag are identical to that of using binary files with forcing flag 3.

1.10.8 Monthly Forcings

*F*ormat:
7 {.mon filename}
{beginning unix time} {ending unix time}

A forcing flag of 7 indicates a uniform in space forcing that recurs monthly. When the end of the calander year is reached, the monthly forcing file (.mon) is read again from the beginning. The beginning unix time is used to determine the month the simulation begins (for this forcing). If the total simulation time takes the simulation past the ending unix time, the forcing is assumed to be 0.0 for all locations and times beyond the ending unix time.

1.11 Dams

Format:
{*dam flag*} *[.dam or .qvs filename]*

This section specifies whether dams will be used. A dam flag of 0 means no dams are used. A flag of 1 indicates a dam file (.dam) will be used, and a flag value of 2 indicates a discharge vs storage file (.qvs) will be used. Some models do not support dams. For these models, the dam flag must be set to 0 or an error occurs.

1.12 State Forcing Feeds

Format:
{*reservoir flag*} *[.rsv or .dbc filename] [forcing index]*

This section specifies whether a forcing provided in the Forcings section is to be used as a forcing of the states of differential or algebraic equations at some links. A reservoir flag of 0 indicates no forcing will by applied to system states. A flag of 1 indicates state forcings will be applied to all link ids in the specified .rsv file. A reservoir flag of 2 indicates state forcing will be applied to all link ids pulled from the database query 0 in the given .dbc file. If the reservoir flag is not 0, then the index of the forcing in the Forcings section must be specified.

1.13 Time Series Location

Format:
{*time series flag*} *[time resolution] [.dat / .csv / .dbc filename] [table name]*

This section specifies where the final output time series will be saved. A time series flag value of 0 indicates no time series data will be produced. Any flag with value greater than 0 requires a time resolution for the data. This value has units equal to the units of *total simulation time* (typically minutes). A value of -1 uses a resolution which varies from link to link based upon the expression

$$\left(0.1 \cdot \frac{A}{1\ km^2}\right)^{\frac{1}{2}}\ min \tag{1}$$

where A is the upstream of the link, measured in km^2.

A time series flag of 1 indicates the results of the simulation will be saved as a .dat file. The filename complete with a path from the binary file must be specified. If a file with the name and path given already exists, it is overwritten. A time series flag of 2 indicates the results will be stored as a .csv file. A time series flag of 3 indicates the results will be uploaded into the database described by the given .dbc. In this case, a table name accessible by the queries in the .dbc file must be specified.

This section is independent of the section for *Link IDs to Save* described below. For example, if link ids are specified in the *Link IDs to Save* section and the *time series flag* in the *Time Series Location* set to 0, no output is generated. Similarly, if the *time series id flag* is set to 0 in the *Link IDs to Save* section and the *time series flag* is set to 1, a .dat file with 0 time series is produced.

Notice: the *time resolution* is entirely independent of the time step used by the numerical integrators. Reducing this value does NOT produce more accurate results. To improve accuracy, reduce the error tolerances described below. There is no built-in way to produce results at every time step. If you need results at every time step, please provide a compelling reason. This is a very easy way to crash a compute node or file system.

1.14 Peakflow Data Location

Format:
{*peakflow flag*} *[.pea / .dbc filename] [table name]*

This section specifies where the final peakflow output will be saved. A peakflow flag of 0 indicates no peakflow data is produced. A peakflow flag of 1 indicates the peakflow results of the simulation will be saved as a .pea file. The filename complete with a path from the binary file must be specified. A peakflow flag of 2 indicates the results will be uploaded into the database described by the given .dbc. In this case, a table name accessible by the queries in the .dbc file must be specified.

This section is independent of the section for *Link IDs to Save* described below. For example, if link ids are specified in the *Link IDs to Save* section and the *peakflow flag* in the *Peakflow Data Location* set to 0, no output is generated. Similarly, if the *peakflow id flag* is set to 0 in the *Link IDs to Save* section and the *peakflow flag* is set to 1, a .pea file with 0 peakflows is produced.

1.15 Link IDs to Save

Format:
{*time series id flag*} *[.sav / .dbc filename]*
{*peakflow id flag*} *[.sav / .dbc filename]*

This section provides the list of link ids in which data is produced. The first line is for the time series outputs, while the second is for the peakflow outputs. The *time series ID flag* and the *peakflow ID flag* take the same list of possible values. A flag of 0 indicates no link IDs for which to produce data. A flag of 1 indicates the list of link IDs is provided by the corresponding save file (.sav). A flag of 2 indicates the list of link IDs is provided by the database specified in the given database connection file (.dbc). A flag of 3 indicates that all links will have data outputted.

Warning: a *time series ID flag* of 3 can easily wreak havoc on a file system for simulations with a large number of links. At the very least, extremely large output files and database tables will occur. Be very careful with this!

This section is independent of the sections for *Time Series Location* and *Peakflow Data Location* above. For example, if link ids are specified in the *Link IDs to Save* section and the *time series flag* in the *Time Series Location* set to 0, no output is generated. Similarly, if the *time series id flag* is set to 0 in the *Link IDs to Save* section and the *time series flag* is set to 1, a .dat file with zero time series is produced.

1.16 Snapshot Information

Format:
{*snapshot flag*} *[.rec / .dbc filename]*

This section specifies where snapshot information is produced. A snapshot is a record of the every state at every link in the network. Snapshots are produced at the end of simulations. This is useful for beginning a new simulation where an old one ended. A snapshot flag of 0 indicates no snapshot is produced. A snapshot flag of 1 indicates the snapshot will be produced as a recovery (.rec) file with path and filename specified. A snapshot flag of 2 indicates the snapshot will be uploaded to the database specified by the database connectivity (.dbc) file.

1.17 Scratch Work Location

Format:
{*filename*}

This section specifies the location of temporary files. These files are used to store intermediate calculations. The filename can include a path name. If the file already exists, the contents are overwritten.

Notice: Be sure to use different filenames for simulations intended to run simultaneously. Two simulations writting into the same scratch files is probably not what you want!

1.18 Error Control Parameters

Format:
{*facmin*} {*facmax*} {*fac*}

This section specifies parameters related to the error control strategy of the numerical integrators. The value *facmin* represents the largest allowed decrease in the stepsize of the integrators as a percent of the current step. Similarly, *facmax* represents the largest allowed increase. The value *fac* represents the safety factor of the integrators. Any accepted stepsize is multiplied by this value. Typically values of *facmin*, *facmax*, and *fac* are 0.1, 10.0, and 0.9, repectively.

1.19 Numerical Error Tolerances

Format:
{*solver flag*} *[.rkd filename]*
[rk solver index]
[absolute error tolerance 1] [absolute error tolerance 2] ...
[relative error tolerance 1] [relative error tolerance 2] ...
[dense absolute error tolerance 1] [dense absolute error tolerance 2] ...
[dense relative error tolerance 1] [dense relative error tolerance 2] ...

This section specifies error tolerances for the numerical integrators. A *solver flag* of 0 indicates the same tolerances will be used for all links. The tolerances are specified later in this section. A *solver flag* of 1 indicates the tolerance info will be specified in the given RK data (.rkd) file.

If *solver flag* is 0, than an *rk solver index* must be specified. !!!! Give table of methods !!!!. Each error tolerance must have a value for each state of the system. The order of the tolerances must match the order of the states in the state vectors. The absolute and relative error tolerances are those typically used for RK methods. The dense tolerances are for the numerical solution produced between time steps. A numerical solution is rejected if either the error tolerances or dense error tolerances for any state is believed to be violated.

2 Model Descriptions

Several models come hard coded with the solvers. The evaluation of the differential and algebraic equations occurs in the source code *problems.c*, while the definition of the models (i.e. number of parameters, precalculations, etc.) is set in *definetype.c*. New models can be added here by modifying those two source files, plus adding needed function declarations to *problems.h*.

Each model has a number of states that are to be determined at each link. A specific time, these states are stored in a vector, known as a *state vector*. Similarly, the value of the differential and algebraic equations at a particular time and state are stored in an *equation-value vector*. There is a correspondence between state and equation-value vectors at each link. For example, if the underlying model is a system of differential equations, then the derivative of the first state in a state vector is stored in the first entry of the equation-value vectors, the derivative of the second state in a state vector is sotred in the second entry of the equation-value vectors, etc.

First, we give an explaination of all the information needed to define a model. After this, a description of the differential and algebraic equations for each model is given, as well as an explaination of the parameters.

Every built in model is given a unique id known as the model type. This nonnegative integer value is used to identify the model throughtout the initialization process. The model type must be specified in the global file used to initialize ASYNCH.

2.1 Model Definition

The definition of every model is given in *definetype.c*. This module consists of five routines used to initialize each model. A description of the contents of each of these routines is given below.

2.1.1 SetParamSizes

This routine defines several integral values needed to describe a model.

dim The number of states modeled by the differential and algebraic equations. This parameter is also the number of such equations at a single link.

template_flag A flag to specify if the model uses an XML parser for evaluating the differential equations. 0 indicates no parser, 1 indicates a parser is used.

assim_flag A flag to specify if the model uses a data assimilation scheme. 0 indicates no data assimilation, 1 indicates data assimilation is used. This feature will be removed in future versions.

diff_start The index in the equation-value vectors where the differential equations begin. All equations before this index are assumed to be algebraic. If all equations are differential, then this value should be 0.

no_ini_start The index in the state vectors corresponding to the first state not requiring initial conditions specified by the initial states specification in a global file. These states are generally initialized by other parameters of the model in the function *InitRoutines*. If all states require initialization through the global file (typical), then no_ini_start should be set to dim.

num_global_params The number of global parameters specified in a global file for this model. If the number provided by the global file is less than expected, an error occurs. If more parameters are given than expected, a warning is given. The extra parameters are accessible. Although providing more parameters than needed is not recommended, it can be useful for testing.

uses_dam A flag to indicate if the model uses dams. 0 indicates no, 1 indicates yes. If dams are not available for this model, a value of 0 is expected for the dam flag in the global file.

params_size The total number of local parameters available at each link. This includes all parameters read from the hillslope parameters of the global file as well as all precalculations (specified in the *Precalculations*).

iparams_size The number of integer valued parameters at each link. This may be removed in future versions.

dam_params_size The number of additional parameters at links with a dam. These parameters are included at the end of the vector of parameters at each link with a dam.

area_idx The index in the parameter vector of the upstream area parameter. This is used to determine an order in which the equations at the links will be integrated. This parameter is also used frequently with peakflow data.

areah_idx The index in the parameter vector of the hillslope area. This parameter is frequently used with peakflow data.

disk_params The number of local parameters available at each link read from a parameter file or database table. The params_size minus the disk_params is the number of precalculated parameters plus any dam parameters at the link.

num_dense The number of states passed down from one link to another. This number cannot be larger than dim. If equal to 0, then the links are disconnected.

convertarea_flag Flag to indicate whether the model converts the parameter with index area_idx from km^2 to m^2. This can be needed for peakflow output data.

num_forcings The number of forcings for the model. If a global file specifies less than this number of forcings, an error occurs. If more than this number of forcings is specified, a warning is given.

dense_indices An array containing the indices in the state vectors that are passed from one link to another. This array must contain num_dense indices.

2.1.2 ConvertParams

This routine allows for unit conversions on the local parameters at each link. These conversions occur immediately after the parameters are loaded into memory, and thus will be in place for all calculations. This feature is useful for when a data source provides values with units different than those expected by the model.

2.1.3 InitRoutines

This routine specifies which routines are associated with the model. In this routine, the following arguments are available.

link The current link where the routines are to be set.

type The model index.

exp_imp A flag to determine if an implicit or explicit RK method is to be used. 0 if the method is explicit, 1 if it is implicit.

dam A flag for whether a dam is present at this link. 0 if no dam is present, 1 if a dam is present.

The following routines must be set at each link.

RKSolver The routine for the numerical integrator.

f The routine to evaluate the differential equations of the model.

alg The routine to evaluate the algebraic equations of the model.

Jacobian The routine to evaluate the Jacobian of the system of differential equations. This must be set if an implicit RK method is used.

state_check The routine to check in what state the system is. The number of the system state is determined by the model. !!!! Rename this. Confusing with unknowns. !!!!

CheckConsistency This routine alters the state vectors to be consistent with constraints of the system. **Notice: these constraints MUST exist in the exact solution of the equations for the link (for example, nonnegative solutions to a linear system).**

2.1.4 Precalculations

This routine allows computations that are static in time and independent of state to be performed. The results are stored with the hillslope parameters. This feature can be used to prevent redundant computations. The following information is accesible in this routine:

link_i The current link where precalculations are performed.

global_params The parameters which are constant in space and time.

params The parameters for this hillslope. Results from this routine will be stored in this vector. Other parameters from a database or parameter file (.prm) are accessible here.

iparams The integral parameters for this hillslope.

disk_params The first entry of params that should be set for this location.

params_size The first entry for dam parameters. These are only accessible if the dam flag is set.

dam The flag to indicate if a dam is present at the current hillslope. If dam is 1, then a dam model is present here. If dam is 0, then a dam model is not present.

type The index of the model.

Before exiting, all entries in params from index disk_params up to (but not including) params_size should be set.

2.1.5 ReadInitData

This routine sets any initial conditions which are determined through *Initial Conditions* section of the global file (.gbl). Generally, this is to set the initial conditions for unknowns in models determined by algebraic equations. The following information is available in this routine:

global_params The parameters which are constant in space and time.

params blah