Documentation for INCABuilder model builders - documentation is in development

Magnus Dahler Norling

February 21, 2019

1 Introduction

This is documentation for model developers. Documentation for model users and framework developers will be provided in separate documents (eventually).

Note 1. It is recommended for any model developers to take a look at the tutorials and experiment with making changes to them before reading all of the documentation. The tutorials will also provide the information about how to compile the models into finished exes.

2 Basic concepts

2.1 The model

The model object contains all immutable information about the model that will not change with each particular usage of the model. The model object contains lists of various model entities and information about these that are provided by the model developer in the model registration routine. The different model entities are presented in Table 2.1

Index sets	Parameters, inputs and equation results can index over one or more index sets. For instance, you may want to evalute the same equations in multiple contexts, such as once per reach in a river, and in that case you want an index set for these reaches.
Parameter groups	Parameter groups are collections of parameters. A parameter group can vary with an index set, making each parameter have a separate value for each index in the index set. The parameter group can also be a child group of another parameter group. In that case, each parameter in the group has a separate value for each pair of indexes from the index set of its group and the parent group. This can be chained to make parameters vary with as many index sets as one wants.
Parameters	Parameters are values used to tune the equations of the modell. Parameters do not vary over time. Currently four different types of parameters are supported: double precision floating point, unsigned integer, boolean and time.
Inputs	Inputs are forcings on the model that vary over time. An example of an input is a time series of daily air temperature that has been measured in the field. Inputs can also vary with index sets. For instance, one can have a separate air temperature series for each subcatchment area in the model.
Equations	The model has a set of equations that are executed in a specific order for every timestep of the model (see later for how the run order is determined). Each equation can look up the value of various parameters, inputs, results of other equations (both from the current timestep and earlier) and produces a single output value for each time it is evaluated. There are several types of equations. The main two are discrete timestep equations and ordinary differential (ODE) equations.
Solvers	Each solver contains a list ODE equations that it will solve over one timestep whenever it is run. The solver must be registered with additional information about e.g. which integrator method it will use.

Table 1: The model entities

2.2 The dataset

The dataset contains information about the specific setup of a model. This includes all the indexes of each index set, as well as all the specific values of the parameters and input series. It will also contain the result series after the model is run. The typical way of setting up a dataset is by reading it in from a parameter file and an input file. The file formats for these are provided in separate documentation.

3 Model structure

3.1 Index sets and indexes

Index sets is one of the fundamental concepts in the INCABuilder framework [Description!!].

Which index sets each model has is fixed, and determined by the model developer. However, which indexes each index set contains can usually be determined by the model user, for instance by configuring a parameter file. Some model do however require a fixed set of particular indexes for some of its index sets. For instance INCA-N only works with three soil boxes, while INCA-C works with four.

3.2 Parameter structure

Each parameter belongs to a parameter group, which again can belong to another parameter group and so on. Moreover, each parameter group can index over an index set. This means that each parameter can in practice be viewed as a multidimensional array of parameter values, where each dimension in the array is indexed by one of the model's index sets. It is possible for a parameter to index over the same index set multiple times. For instance, PERSiST has a "Percolation matrix" parameter where both the row and column in the matrix is indexed by the soil box.

Example 1. A PERSiST "Percolation matrix", from a parameter file. (See separate documentation for detailed description of the parameter file format).

```
"Percolation matrix" :
0.2 0.8 0.0
1.0 0.5 0.5
0.0 0.0 1.0

0.1 0.9 0.0
1.0 0.6 0.4
0.0 0.0 1.0
```

Here the parameter indexes over "Landscape units" (signifying what matrix you are looking at) and "Soils" twice (signifying which row and column). The percolation matrix is used says something about what proportion of the water will flow between each soil box, but we will not go into detail about that here.

The parameter group and index set structure of the parameters are entirely fixed by the model (developer), and can not be configured in the parameter file (though one can usually configure which indexes each index set has).

3.3 Input structure

An input is a timeseries (usually of measured data or preprocessed based on measured data). Inputs are used as forcings in many models. For instance, hydrological models often use precipitation timeseries to determine how much water enters the system at each day. Inputs can either be global for the system, or like parameters can index over one or more index sets. For instance, one may want to have a separate precipitation timeseries for each subcatchment (reach). Unlike with parameters, which index set each input indexes over is determined in the input file. This can sometimes change the equation structure of the model. For instance, if precipitation and air temperature does not depend on which subcatchment one are in, the framework may decide for some models that equations like "Precipitation falling as snow" should only be evaluated once globally instead of per subcatchment. This does however also depend on which parameters these equations reference. If "Precipitation falling as snow" references a parameter that indexes over the "Reaches" index set, that equation will still evaluate once per reach.

3.4 Equation batch structure

During the main run of the model, the model will for each timestep evaluate all equations (sometimes called state variables) in a given order. Equations can also be evaluated for multiple

indexes. For instance the equations having to do with flow or volume of the reach are evaluated for each reach index. Equations are sorted so that an equation is always evaluated after other equations that it uses the result values of. To facilitate this, the model builds an equation batch structure.

Example 2. The equation batch structure for PERSiST.

```
**** Result Structure ****
[Reaches][Landscape units]
       -----
       Snow melt
       Rainfall
       -----

[Reaches][Landscape units][Soils]
       -----
       Percolation input
       Saturation excess input
       Input
       Water depth 1
       Saturation excess
       Water depth 2
       Evapotranspiration X3
       Evapotranspiration X4
       Evapotranspiration
       Water depth 3
       Total runoff
       Drought runoff
       Percolation out
       Water depth 4
       Runoff
       Runoff to reach
       Water depth
       -----

[Reaches][Landscape units]
       -----
       Snow fall
       Snow as water equivalent
       (Cumulative) Total runoff to reach
       Diffuse flow output
       -----

[Reaches]
       -----
       (Cumulative) Total diffuse flow output
       Reach flow input
       ----- (SOLVER: Reach solver)
       Reach time constant
       (ODE) Reach flow
       (ODE) Reach volume
       -----
       Reach velocity
       Reach depth
       -----
```

The model has an ordered list of batch groups, where each batch group has an ordered list of batches. Each batch group also has an ordered list of index sets that it indexes over. Each

batch can either be a discrete batch or a solver batch. The model will evaluate each batch group at a time. Then for each combination of indexes in the index sets it evaluates each batch in the batch group in order. In discrete batches, each equation is evaluated in order. In solver batches, the ODE system will be integrated possibly evaluating each equation many times. In Example 2, the first three batch groups contain one batch each, while the last batch group contains three batches.

Note 2. The only time a batch group can have more than one batch is if it contains both a solver batch and one or more discrete batches (or possibly another solver batch using a different integrator). Two discrete batches next to each other that index over the same index sets would have been merged.

Example 3. Pseudocode for how the model evaluates the batch structure in Example 2.

```
for every timestep:
       for every reach R:
              for every landscape unit LU:
                     evaluate "Snow melt" and "Rainfall" for this LU in this R.
       for every reach R:
              for every landscape unit LU:
                     for every soil box S:
                            evaluate "Percolation input" to "Water depth" for this S
                                 in this LU in this R
       for every reach R:
              for every landscape unit LU:
                     evaluate "Snow fall" to "Diffuse flow output" for this LU in
                         this R
       for every reach R:
              evaluate "Total diffuse flow output" and "Reach flow input"
              run a solver that integrates the ODE system containing the equations "
                  Reach time constant" to "Reach volume" (possibly evaluating them
                  many times)
              evaluate "Reach velocity" and "Reach depth"
```

At the end of a model run, one can extract a timeseries of each of these equations for any combination of indexes that they index over. For instance one can look at the timeseries for "Snow melt" in {"Langtjern", "Forest"}, or the timeseries for "Water depth" in {"Langtjern", "Peatland", "Organic soil layer"} (given that these were indexes provided for this particular dataset).