Fors

Recommended version: n205

Always check that the version given here is the latest. Always use the most recent version of the program(s), located in G:\Hytools\swax

Licence:

> “
> Your use of this software indicates your agreement to the following licence:
>
>
> ”

Wishlist

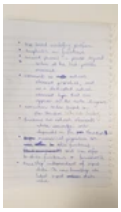
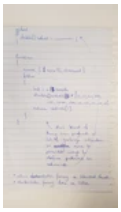
- [ ] Support inflow bias NE 1 in PWL routing.
- [ ] DREAM algorithm.
- [x] Backward-Euler method on storages.
- [ ] New [recorder_exporter] section that specifies a multi-column csv where all recorders are exported. They should appear in the order they're listed in the fors file, and the column names should be similar to how they appear in ForsCalibrate, e.g. "mynode\0". Once this is done, the next wish would be to support pixie.
- [ ] Custom expression to have double-mass function. What is the relevant statistic here?
- [ ]  
- [x] @Chas Egan (DES), can you please explain what happens when you run Fors model with residual flow that has negatives? Looks like it sets negative inflows to zero (Alma).

Table of Contents

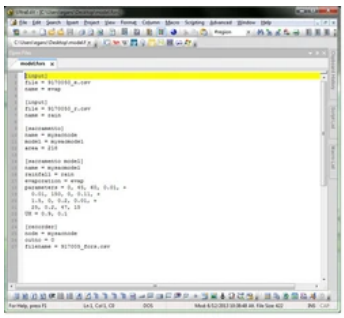
- 1 Wishlist
- 2 Table of Contents
- 3 Introduction
- 4 How to build a fors model
  - 4.1 [Input]
  - 4.2 [Node]
  - 4.3 [Inflow]
  - 4.4 [Gauge]
  - 4.5 [DMM Target]
  - 4.6 [Loss]
  - 4.7 [Nonlinear reach]
  - 4.8 [PWL reach]
  - 4.9 [Timeseries flux]

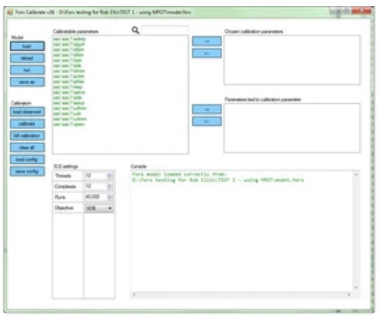
        - 4.10 [Function]
        - 4.11 [Splitter]
        - 4.12 [Aquifer Recharge]
        - 4.13 [StorageBackwardEuler]
        - 4.14 [Storage]
        - 4.15 [Lag reach]
        - 4.16 [GR4J]
        - 4.17 [GR4JSG]
        - 4.18 [GRSGDPIE]
        - 4.19 [AWBM]
        - 4.20 [Sacramento]
        - 4.21 [Sacramento model]
        - 4.22 [Recorder]
- 5 Calibrating a fors model
    - 5.1 Target timeseries data
    - 5.2 Calibratable parameters
        - 5.2.1 Sacramento parameter ranges
        - 5.2.2 GR4J parameter ranges
        - 5.2.3 Nonlinear routing parameter ranges
    - 5.3 Objective functions
        - 5.3.1 Inbuilt objective functions
        - 5.3.2 Custom objective function
            - 5.3.2.1 Example custom function expressions
        - 5.3.3 Optimization algorithms
- 6 Running a fors model from the commandline
- 7 Using ForsGaugeDataPrep
- 8 Example 1: Headwater Sacramento
- 9 Example 2: Residual Reach
- 10 Reference Material

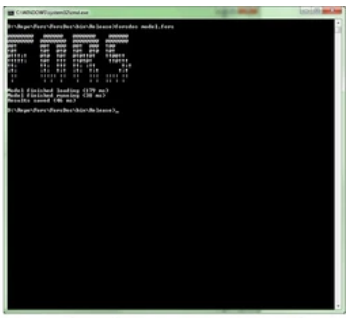
Introduction

[Swedish; A fast-flowing and turbulent part of the course of a river. Pronounced “fosh”]

“Fors” is a simple hydrologic modelling platform, which can be used to build and run simple hydrologic models. Its primary purpose is for calibrating Source models. To calibrate a Source model using fors, the user builds a replicate of their Source model (or part thereof) within fors and calibrates it with the fors calibration tool. The calibrated parameters and data can then be used in the original Source model.


fors model


fors calibration tool


running from commandline

Fors offers only a subset of Source functionality, limited to functionality required in a calibration model. This includes rainfall-runoff modelling, streamflow routing, losses, storages, inflows, extractions, confluences, and anabranches. It emulates Source in the way these processes are modelled, making it suitable for building and calibrating hydrologic models that can then be ported directly to Source.

The advantages of fors are:

- Performance. Each fors models runs about 100x faster than its Source equivalent. Fors models can easily be run in parallel, and the fors calibration tool takes advantage of this to achieve overall performance over 1000x faster than Source.
- Smarter calibration objective functions. The objective functions have been developed to achieve balanced performance in several areas: daily flow (both high and low regimes), total volume, and exceedance curves. There is still room for improvement, but we have come a long way from the old NSE or NSELog objective functions.
- Fors uses a text-based model file format making it easy to save, edit, share, and compare models.
- The calibration tool uses a parallelized Shuffled Complex Evolution (SCE; Duan etal 1992) optimization algorithm.

Development of fors started in early-2013, building upon work done for the Flinders and Gilbert Agricultural Resource Assessment (FGARA) project in 2012.

How to build a fors model

Fors models are text based and can be built in any text editor (e.g. UltraEdit). A fors model consists of input data, Sacramento models, nodes, and recorders for outputting model results. Square brackets “[…]” are used to denote the start of a definition of some component:

- a timeseries input - “[input]”
- a node - “[gauge]”, “[inflow]”, “[lag reach]”, “[loss]”, “[node]”, “[nonlinear reach]”, “[sacramento]”, “[storage]”, “[splitter]”, or “[timeseries flux]”
- a Sacramento rainfall runoff model - “[sacramento model]”
- a result recorder - “[recorder]”

Here is a simple example to get us started:

```
[input]
file = 9170050_e.csv
name = evap

[input]
file = 9170050_r.csv
name = rain

[sacramento]
name = mysacnode
model = mysacmodel
area = 218

[sacramento model]
name = mysacmodel
rainfall = rain
evaporation = evap
parameters = 0, 45, 60, 0.01, +
  0.01, 150, 0, 0.11, +
  1.5, 0, 0.2, 0.01, +
  25, 0.2, 47, 15
UH = 0.9, 0.1

[recorder]
node = mysacnode
outno = 0
filename = 917005_fors.csv
```

This example model has two input timeseries (some rainfall and evaporation data). There is just one node defined: a Sacramento node named “mysacnode”. There is a Sacramento model named “mysacmodel”. And finally there is one recorder that saves the modelled flow downstream of “mysacnode” to a file called “917005_fors.csv”.

The model file is not case-sensitive. Comments can be made inside the model file. They should be preceded by “//”. The continuation character “+” can be used to carry on the definition of some parameter to a new line.

[Input]

An input timeseries is declared by starting with the “[input]” tag and then providing a name and source file for the data. Once defined, the dataset can be referenced anywhere in the model that requires an input timeseries.

Arguments	Arg Type	Value
Name	Compulsory	Name for the data series. This must be unique. E.g. Name = GaugedInflows
File	Compulsory	Name of the file to load data from. Refer to the section on timeseries data file formats. E.g. File = inflow.csv
Column	Optional	(It is conventional to not use this) This parameter lets you to choose a column from a multi-column CSV file. If you provide an integer (0, 1, 2, 3..) then Fors will choose the column by counting them with 0 being the one immediately following the dates. But if you give a value that isn't an integer (e.g. column = 031029 Rain) then it will look for the first exact match (not case-sensitive). E.g. Column = daily rainfall (mm)

Example input definition with separate files:

```
[input]
name = 031029_rain
file = 031029_rain.csv

[input]
name = 031038_rain
file = 031038_rain.csv

[input]
name = 031029_mwet
file = 031029_mwet.csv
```

or a similar thing reading data from a multi-column data file

```
[input]
name = 031029_rain
file = climate_data.csv
column = 031029 Rain (mm)

[input]
name = 031038_rain
file = climate_data.csv
```

```
column = 031038 Rain (mm)

[input]
name = 031029_mwet
file = climate_data.csv
column = 031029 Mwet (mm)
```

[Node]

The generic node is declared by starting with “[node]” tag and then providing a name and defining downstream links. It has no particular functionality but (like any node) can function as a confluence.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyBasicNode
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

The generic node has just one recordable timeseries, being the total outflow [ML/day] from the node.

OutNo	Description
0	Total node downstream flow [ML/day]

[Inflow]

An inflow node is declared using the “[inflow]” tag. An inflow node adds water to the river system.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyInflowNode
Inflow	Compulsory	Name of the input dataset containing the inflow data [ML/day]. E.g. Inflow = GaugedInflows
Factor	Optional	Scaling factor that can be used to scale the inflow timeseries. If not supplied, the factor is internally taken to be 1.
InfillingValue	Optional	If supplied, this value will be used to infill missing values in the input timeseries. CAUTION!!! This will allow this inflow node to execute for all time, and the model simulation period will be determined by other parts of the model. Before using this, think about what this infilling means for the intended purpose of your model.
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

OutNo	Description
0	Total node downstream flow [ML/day]
1	Net influx in this node [ML/day]

[Gauge]

A gauge node is declared using the “[gauge]” tag. A gauge node can be used to compare modelled and observed data, or override modelled flows with historic observed flows at this point in the model.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyInflowNode
Observedflow	Optional	Name of the input dataset containing the observed flows [ML/day]. These can be used to override modelled flows at a point (see ApplyObserved) or for calculating the difference, observed – modelled. E.g. Observedflow = GaugedInflows Note that negative values in the "Observedflow" data are treated as missing values. If observed values are being applied (ref ApplyObserved), no changes will be made to the model flow on days when the observed data is missing or negative.
ApplyObserved	Optional	Replaces the modelled flows at this point in the model with observed flows on the days they are provided.

Arguments	Arg Type	Value
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

OutNo	Description
0	Total node downstream flow [ML/day]
1	Total upstream flow [ML/day]
2	Observed minus modelled [ML/day]

[DMM Target]

A DMM Target node is declared using the “[dmm target]” tag, and is used to indicate the target site (and parameters) for a Fors DMM inflow adjustment. This node has no effect in a normal model run, or in a calibration.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyDmmTargetNode
TargetFlow	Optional	Name of the input dataset containing the DMM target flows [ML/day] (i.e. these are usually the observed flows at the gauge). E.g. TargetFlow = MyObservedFlowData
Iter	Optional	This is an optional integer to specify the number of iterations in your DMM run. If you don't specify a value, it will default to 10 iterations. E.g. Iter = 5 Note: You can use ForsDesigner's DMM console output to monitor the adjustments (size and number of adjustments) from one iteration to the next. This can be helpful if you're wondering how many iterations to do. For many simple cases you may only need 1 or 2 iterations. For complicated systems with much nonlinear routing you may need 10 or more iterations. Please note that MORE IS NOT BETTER, as in some cases additional iterations (beyond some point) can continue to modify inflows for very little benefit at the target gauge. Please refer to the DMM page for more guidance.
ScalingThreshold	Optional	The scaling threshold is an optional parameter that takes one or more numerical values to control how DMM apportions adjustments. E.g. ScalingThreshold = 10000000, 0, 0, 0. When the flows are smaller than the scalingthreshold, adjustments will be made by adding/subtracting to the existing inflows in proportion to the inflow weights. When the flows are greater than the scalingthreshold, adjustments will be made by scaling up/down the existing inflow values. When more than one value is provided, the different values will be used in different iterations. In the example above the first iteration will have a scalingthreshold of 10000000 and subsequent iterations will have a scalingthreshold of 0. Refer to the DMM page for more details. If no scalingthreshold is defined, fors will use a default value of 0. I.e. inflows will always be scaled, unless the flows are zero.
AdjustmentOutputFilename	Optional	Here you have the option to provide a filename for DMM to output the adjustments. This is a multi-column csv file and can be useful in testing and reporting. E.g. AdjustmentOutputFilename = dmm_adjustments.csv
Weights	Optional	The optional weights parameter can be used to specify the weights (one weight per inflow), which DMM will use when adjusting by adding/subtracting flow. This happens when the flow is less than the scalingthreshold (including when the flow is 0). The format for specifying weights is a comma-delimited sequence of key: value pairs. E.g. Weights = Reach_J1_predmm: 0.1783, Reach_J2_predmm: 0.0335, Reach_J3_predmm: 0.00172, Reach_j4_Sac: 0.7708 Like any other parameter in Fors, you can spread the weights definition over multiple lines using a '+' as a continuation character, e.g. Weights = Reach_J1_predmm: 0.1783, + Reach_J2_predmm: 0.0335, + Reach_J3_predmm: 0.00172, + Reach_j4_Sac: 0.7708 Note: if you don't specify the weights, Fors will just use the mean inflow (e.g. your Sacramento flows) at each inflow node to define the corresponding weight. This is typically fine.
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

OutNo	Description
0	Total node downstream flow [ML/day]
1	Target flow [ML/day]
2	Downstream flow minus target flow [ML/day]

Example node definition:

```
[dmm target]
name = dmm_target_112004a
targetflow = gs112004a_observed_flow
weights = Reach_J1_predmm: 0.1783, +
          Reach_J2_predmm: 0.0335, +
          Reach_J3_predmm: 0.00172, +
          Reach_j4_Sac: 0.7708                          //OPTIONAL
scalingthreshold = 100000000, 0                         //OPTIONAL
iter = 3                                                //OPTIONAL
adjustmentoutputfilename = dmm_inflow_adjustments.csv   //OPTIONAL
dsnode = gauge_112004a
```

[Loss]

An loss node is declared using the “[loss]” tag. An loss node uses a loss table (flow-vs-loss) to lose water from the river system.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyInflowNode
LossTable	Compulsory	Name of the file containing the loss table. The first column should be flow [ML/day], and the second column should be loss [ML/day]. Refer to the section on table file formats. E.g. Losstable = losses.csv
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

OutNo	Description
0	Total node downstream flow [ML/day]
1	Net influx in this node [ML/day]
2	Loss volume [ML/day]

Example node definition:

```
[loss]
name = mylossnode
losstable = loss.csv
dsnode = mynextnode
```

[Nonlinear reach]

A nonlinear reach is modelled in fors using a node declared with the “[nonlinear reach]” tag. It is used to introduce Laurenson's (2-parameter nonlinear) routing.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyInflowNode
Length	Optional	Length of the reach [km]. Defaults to 1. E.g. Length = 95.
Parameters	Compulsory	There are two parameters. (1) The first parameter is the Laurenson storage constant K. If the reach length is not specified, then the units of K are [m^(3-3mu) s^mu]. If the reach length is specified, then the units of K are [m^(3-3mu) s^mu/km]. (2) The second parameter is the unitless storage exponent mu. E.g. Parameters = 86400, 0.75.
Area	Optional	Optional constant surface area for rainfall and evaporation calcs [km2]. Default value is 0. E.g. Area = 24.
Rainfall	Optional	Name of dataset containing rainfall data [mm]. E.g. Rainfall = MyRainfallData
Evaporation	Optional	Name of dataset containing evaporation data [mm]. E.g. Evaporation = MyEvaporationData
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

OutNo	Description
0	Total node downstream flow [ML/day]

OutNo	Description
1	Reach storage volume [ML]
2	Reach surface area [km2]
3	Reach surface evaporation losses [ML/day]
4	Reach surface rainfall gains [ML/day]

[PWL reach]

A piecewise-linear reach (Bigmod routing reach) is modelled in fors using a node declared with the “[pwl reach]” tag. In this implementation the reach inflow is used as the reference flow - this is the same as Bigmod, and can be achieved in Source by setting the "inflow bias" parameter x=1.

WARNING! The method is only generally stable if the number of divisions is larger than the maximum travel time.

Surface evaporation and rainfall can be modelled. The current implementation is assumes the surface area is constant with flow. The evaporation may be lower than expected (area * evap_mm) if any of the divisions within the reach dry up, including if the whole reach dries up.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyInflowNode
NDivisions	Optional	Number of reach subdivisions. **WARNING!** The method is only generally stable if the number of divisions is larger than the maximum travel time. If this parameter is omitted (or < 0), the number of divisions will be automatically calculated as follows. NDivisions = Ceiling(maximum travel time) * 2.
~~Length~~	~~Optional~~	~~**You shouldn't use this, or if you do have it explicitly defined in your model, then you should use the default value, Length=1.** If you think you should use a non-default length make sure you talk to Chas first. Using this parameter changes the meaning of the PWL table such that dependent variable is travel time per km (instead of travel time). This optional functionality has some use when automatically calibrating multiple reaches together, e.g. you could use it to tie per-length routing parameters during the calibration. When calibrating in this per-length mode, the optimiser may change the value of NDivisions in accordance with it's in-bulit nominal maximum travel time of 0.0694 m/s (5 days per 30 km).~~ This funcitonality is redundant and no longer supported as of version n110.
Area	Optional	Optional constant surface area for rainfall and evaporation calcs [km2]. Default value is 0. E.g. Area = 24.
Rainfall	Optional	Name of dataset containing rainfall data [mm]. E.g. Rainfall = MyRainfallData
Evaporation	Optional	Name of dataset containing evaporation data [mm]. E.g. Evaporation = MyEvaporationData
PWLTable	Compulsory	A sequence of comma delimited values representing the "Reference Flow [ML/d]" - vs - "Travel Time [days]" table.
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

OutNo	Description
0	Total node downstream flow [ML/day]
1	Reach storage volume [ML]
2	Reach surface area [km2]
3	Reach surface evaporation losses [ML/day]
4	Reach surface rainfall gains [ML/day]

Example node definition:

```
[pwl reach]
name = mypwlreach
ndivisions = 10
area = 24.0
evaporation = myevap
rainfall = myrain
pwltable = +
    0, 2.56046797179132, +
    4000, 1.71545773479733, +
    15000, 2.0380146781953, +
    20000, 2.28167430574076, +
```

```
    31000, 3.06146824921292, +
    45000, 3.72441895215514, +
    57500, 3.57518334955488, +
    71000, 2.34414961295908, +
    120000, 1.07027207398824, +
    200000, 1.45438133537028
dsnode = corowagauge
```

[Timeseries flux]

A timeseries flux node is declared using the "[timeseries flux]" tag. It can be used to add water to the river (an influx) or remove water from the system (an outflux). The timeseries flux node can be used to model historic extractions in fors.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyInflowNode
Influx/Outflux	Compulsory	User must specify either "influx" or "outflux" but not both. This parameter specifies the name of the timeseries data to be applied as a model influx or outflux. E.g. Influx = MyInflowData or Outflux = MyExtractionData
Carryover	Optional	A boolean value TRUE or FALSE, specifying whether unsatisfied extractions should be carried over to the next timestep. The default value is FALSE. E.g. Carryover = true
CarryoverResetMonth	Optional	If you define the CarryoverResetMonth and CarryoverResetDay, then the carryover balance will be reset to 0 every year on the specified day/month. If you define the CarryoverResetMonth but no CarryoverResetDay, then the carryover balance will be reset to 0 every year on the 1st of the specified month. Valid month values are 1 to 12 (where 1 means January).
CarryoverResetDay	Optional	Please read the description for "CarryoverResetMonth". Additionally if you define CarryoverResetDay but no CarryoverResetMonth, then the carryover balance will be reset to 0 on that day of every month (i.e. use CarryoverResetDay=1 to write off the balance coming into each month). Valid day values are 1 to 31.
MaxExtraction	Optional	This optional parameter limits the daily extraction to a given maximum value. This feature works regardless of whether Carryover is applied. If carryover is applied, then demands denied by the application of this limit are carried over to the next timestep.
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

OutNo	Description
0	Total node downstream flow [ML/day]
1	Requested outflow [ML/day]
2	Outflux [ML/day]
3	Denied outflux [ML/day] - the volume of today's original extraction which was not possible today
4	Outstanding requested outflux [ML/day] - the total outstanding carryover at the end of the day

Example timeseries flux node definition:

```
[timeseries flux]
name = my_node_name
outflux = demand_data
carryover = true
carryoverresetmonth = 7
carryoverresetday = 4
maxextraction = 500
dsnode = eos
```

[Function]

A function node evaluates an function expression each timestep when the node is executed. The values may recorded (or used as a calibration target), or optionally the node may be configured to apply those values as a flux on the flows in the network.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyInflowNode

Arguments	Arg Type	Value
Expression	Optional	The expression to evaluate. Example 1: "if(and(month>10,month<3),100,0)" would be 100 in Nov, Dec, Jan, and Feb, and 0 in all other months. Example 2: "0.5*(node1\0(-1) + node1\0(-2))" would calcualte the average of the previous two days' flows at node1. Note that "node1\0(-1)" refers to the value of recorder no 0, of "node1", one day ago (-1), and it will only work if that recorder has been set. The expressions here use the same syntax as calibration expressions and custom objective functions, so the documentation here may be helpful.
ApplyFlux	Optional	A boolean flag. If the value is "false" (the default) the node will not modify the flows, and the downtream flow will be equal to the upstream flow. If the values is "true", the values calculated from the expression will be applied as a flux to the model flows. Positive values will be added to the flow, like an inflow. Negative values will be subtracted from the flow, like an extraction.
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

OutNo	Description
0	Total node downstream flow [ML/day]
1	Total node upstream flow [ML/day]
2	Function value
3	Local flux applied [ML/day]
4	Local flux denied (i.e. any negative flux which cannot be applied because there is insufficient flow) [ML/day]

Example timeseries flux node definition:

```
[function]
name = my_function_node
expression = if(isnan(my_function_node\2(-1)),0,my_function_node\2(-1)+100)
applyflux = true
dsnode = eos
```

[Splitter]

A splitter node is declared using the "[splitter]" tag and uses a table to split the flow between multiple downstream nodes. The splitter may have any number of downstream nodes you choose.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyInflowNode
SplitTable	Compulsory	A sequence of comma delimited values representing the split table. The table specifies how much flow is passed down one link (the "Primary" pathway), while whatever remains is passed down the other link (the "Secondary" pathway). The table also specifies ***which*** downstream link is the primary pathway by naming the corresponding downstream node. Fors uses linear interpolation between the tabulated values (as per Source and IQQM) and if necessary it will extrapolate beyond the end of this table by using a constant ratio based on the last line in the table. Note: this is intended to replace the old "Table" argument below.
~~Table~~	~~Compulsory~~ Depricated	~~Name of the file containing the splitter table. The first column in the table should be flow [ML/day]. Subsequent columns specify the relative proportions of the flow to be propagated to each of the downstream reaches. If there are N downstream nodes, the splitter table should have N+1 columns. Use the downstream node names for the column headers. E.g. Table = splitter_table.csv~~ Still works, but will eventually be phased out. Please use "splittable" instead.
DSNode	Optional	Name of the downstream node. A maximum of 2 downstream nodes are allowed.

OutNo	Description
0	Total node downstream flow [ML/day]

Example node definition:

```
[splitter]
name = mysplit
```

```
splittable = +
    inflow,    lag1, +
    0,         0, +
    60000,     60000, +
    100000000, 60000
dsnode = lag1
dsnode = lag2
```

In the example above the first 60000 ML/day all goes to the node "lag1", and above that everything goes to the node "lag2".

Example split_table.csv **(Depricated - Retained for legacy reference only. Please use "splittable" instead.)**

```
flow [ML/day], lag1,  lag2
0,             0,     0
60000,         60000, 0
100000000,     60000, 99940000
```

[Aquifer Recharge]

An aquifer recharge node is a special type of splitter, which can be used to divert water from the river to a storage (representing an aquifer) based on the volume of water in the storage.

The aquifer recharge node is designed to have two downstream nodes, one of which must be a storage (the aquifer). A lookup table is used to determine the aquifer demand, depending on the volume of water in the storage. The volume of water diverted to the aquifer will be equal to **MIN(aquifer demand, flow in stream)**.

This node was designed for use in development of the Lockyer (Moreton) model but may be of use elsewhere. The setup can be implemented in Source by using a splitter node, with a minimum flow node with a function (ask Cia, Georgia or Chas).

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyNode
DefineRechargeUsingFlowFraction	Optional	This parameter is a boolean flag. If it is set to 'false' (the default value) the aquifer demand is defined in terms of a flow [ML/day], but if it is 'true' the aquifer demand is defined in terms of a fraction of the total node inflow. E.g. DefineRechargeUsingFlowFraction = True
Table	Compulsory	Name of the file containing the aquifer demand table. The first column in the table should be aquifer storage volume [ML]. Second column should be the aquifer demand defined in terms of flow [ML/day] or as a fraction of inflows (no units) (refer to parameter above). E.g. Table = aquifer_demand_table.csv
DSNode	Optional	Name of the downstream node. A maximum of 2 downstream nodes are allowed.

OutNo	Description
0	Total node downstream flow [ML/day]. This includes flow to the aquifer + flow to the stream, and is equal to the node inflow.
1	Aquifer demand [ML/day]
2	Aquifer flow [ML/day]

[StorageBackwardEuler]

A storage node is used to represent a dam, weir, or waterhole.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyStorageNode
Initialstorage	Optional	Initial storage volume [ML]. Unless the storage has been explicitly initialised the 1st time step is not considered valid. If using Fors for a SID the SidFlux will not be returned
Storagetable	Compulsory	Name of the file containing the storage table. The storage table should contain three columns: level [m], storage [ML], surface area [ha]. The storage table must not be exceeded during the simulation. E.g. Storagetable = storagetable.csv
Spillwaytable	Compulsory	Name of the file containing the spillway table. The spillway table should contain two columns: level [m], outflow [ML/day]. The spillway table must not be exceeded during the simulation. E.g. Spillwaytable = spillwaytable.csv
Rainfall	Optional	Name of a dataset containing rainfall data [mm]. E.g. Rainfall = MyRainfallData
Evaporation	Optional	Name of dataset containing evaporation data [mm]. E.g. Evaporation = MyEvaporationData
Seepage	Optional	Constant seepage rate [mm/day]. E.g. Seepage = 0.821

Arguments	Arg Type	Value
Releases	Optional	Name of dataset containing release data [ML/day]. Releases represent regulated releases from the storage to the downstream node. The actual storage outflows in the model may be lower than the specified releases if the storage becomes empty, and may exceed the specified releases if the storage spills. E.g. Releases = MyReleaseData
Extractions	Optional	Name of dataset containing extraction data [ML/day]. Extractions represent usage from the pond. This water is removed from the system and does NOT flow to the downstream node. E.g. Extractions = MyExtractionData
ForcedLevels	Optional	Name of dataset containing observed level data [m]. If this input is supplied, an additional flux (inflow or extraction, called the "SID flux") will be applied to the storage each timestep as required in order to achieve the observed level at the end of each timestep. The main purpose for this is for Storage Inflow Derivation (SID). A note on the Forced Volume recorder: On days when the storage level has been forced using "ForcedLevels", the corresponding storage can be recorded using recorder 11 "Forced volume"; On days when the level has not been forced, recorder 11 "Forced volume" will return a missing value. Note that when using this to do a SID you probably should raise the spillway and generate the observed outflows using the "Releases" argument, since your approximated spillway may prevent the storage from being able to reach the observed level.
Established	Optional	Timeseries that flags whether the storage exists or not. Use a missing value (blank in csv) to indicate that the storage does not exist, and any real value (e.g. 1) to indicate that it does exist. When a storage does not exist, any previously stored volumes are completely spilt, and any new inflows are passed straight through the node. E.g. Established = MyEstablishedFlagData
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

Recorders:

OutNo	Description
0	Total node downstream flow [ML/day]
1	Storage [ML]
2	Surface area [ha]
3	Level [m]
4	Evaporation outflux [ML/day]
5	Rainfall influx [ML/day]
6	Releases [ML/day]
7	Spills [ML/day]
8	Extractions [ML/day]
9	Seepage [ML/day]
10	SID flux [ML/day]
11	Forced volume [ML/day]

Example:

```
[StorageBackwardEuler]
Name = neilturner
InitialStorage = 1467
StorageTable = storage_table.csv
SpillwayTable = spillway_table.csv
Rainfall = Rain
Seepage = 0.821
Evaporation = EvapSeepage
DSNode = out
```

[Storage]

A storage node is used to represent a dam, weir, or waterhole. A storage node is defined using the ”[storage]” tag. Many processes occur simultaneously on a storage including: accumulation of streamflow from upstream nodes, rainfall and evaporation on the ponded area, seepage below the ponded area, uncontrolled spills, and controlled releases.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyStorageNode
Initialstorage	Optional	Initial storage volume [ML]. Unless the storage has been explicitly initialised the 1st time step is not considered valid. If using Fors for a SID the SidFlux will not be returned

Arguments	Arg Type	Value
Storagetable	Compulsory	Name of the file containing the storage table. The storage table should contain three columns: level [m], storage [ML], surface area [ha]. The storage table must not be exceeded during the simulation. E.g. Storagetable = storagetable.csv
Spillwaytable	Compulsory	Name of the file containing the spillway table. The spillway table should contain two columns: level [m], outflow [ML/day]. The spillway table must not be exceeded during the simulation. E.g. Spillwaytable = spillwaytable.csv
Rainfall	Optional	Name of a dataset containing rainfall data [mm]. E.g. Rainfall = MyRainfallData
Evaporation	Optional	Name of dataset containing evaporation data [mm]. E.g. Evaporation = MyEvaporationData
Seepage	Optional	Constant seepage rate [mm/day]. E.g. Seepage = 0.821
Releases	Optional	Name of dataset containing release data [ML/day]. Releases represent regulated releases from the storage to the downstream node. The actual storage outflows in the model may be lower than the specified releases if the storage becomes empty, and may exceed the specified releases if the storage spills. E.g. Releases = MyReleaseData
Extractions	Optional	Name of dataset containing extraction data [ML/day]. Extractions represent usage from the pond. This water is removed from the system and does NOT flow to the downstream node. E.g. Extractions = MyExtractionData
ForcedLevels	Optional	Name of dataset containing observed level data [m]. If this input is supplied, an additional flux (inflow or extraction, called the "SID flux") will be applied to the storage each timestep as required in order to achieve the observed level at the end of each timestep. The main purpose for this is for Storage Inflow Derivation (SID). A note on the Forced Volume recorder: On days when the storage level has been forced using "ForcedLevels", the corresponding storage can be recorded using recorder 11 "Forced volume"; On days when the level has not been forced, recorder 11 "Forced volume" will return a missing value. Note that when using this to do a SID you probably should raise the spillway and generate the observed outflows using the "Releases" argument, since your approximated spillway may prevent the storage from being able to reach the observed level.
Established	Optional	Timeseries that flags whether the storage exists or not. Use a missing value (blank in csv) to indicate that the storage does not exist, and any real value (e.g. 1) to indicate that it does exist. When a storage does not exist, any previously stored volumes are completely spilt, and any new inflows are passed straight through the node. E.g. Established = MyEstablishedFlagData
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

Recorders:

OutNo	Description
0	Total node downstream flow [ML/day]
1	Storage [ML]
2	Surface area [ha]
3	Level [m]
4	Evaporation outflux [ML/day]
5	Rainfall influx [ML/day]
6	Releases [ML/day]
7	Spills [ML/day]
8	Extractions [ML/day]
9	Seepage [ML/day]
10	SID flux [ML/day]
11	Forced volume [ML/day]

Example:

```
[Storage]
Name = neilturner
InitialStorage = 1467
StorageTable = storage_table.csv
SpillwayTable = spillway_table.csv
Rainfall = Rain
Seepage = 0.821
Evaporation = EvapSeepage
DSNode = out
```

[Lag reach]

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyInflowNode
Lag	Compulsory	Integer number of days to lag the flow. E.g. Lag = 2
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

OutNo	Description
0	Total node downstream flow [ML/day]
1	Reach storage [ML]

[GR4J]

The GR4J node implements the GR4J rainfall-runoff model. There are 4 intrinsic parameters, and additional weighting parameters.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyGr4jNode
Area	Compulsory	The area of the catchment represented by this node [km2]. E.g. Area = 123.4
Rainfall	Compulsory	An expression specifying the rainfall for the model. This can refer to a single dataset (e.g. Rainfall = MyRainData) or a linear combination of datasets (e.g. Rainfall = 0.6*MyRainDataA + 0.4*MyRainDataB). Rainfall data should be in mm.
Evaporation	Compulsory	Name of dataset containing evaporation data [mm]. E.g. Evaporation = MyEvaporationData
Parameters	Compulsory	The four GR4J parameters: X1, X2, X3, X4.
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

Recorders:

OutNo	Description
0	Total node downstream flow [ML/day]
1	Runoff volume [ML/day]
2	Rainfall [mm/day]
3	Evaporation [mm/day]
4	Production Store [mm]
5	Routing Store [mm]

Example:

```
[GR4J]
name = mygr4jnode
area = 3821.2
rainfall = Rain
evaporation = evap
parameters = 150, -3, 210, 1.5
dsnode = out
```

[GR4JSG]

The GR4JSG node implements GR4J + Snow & Glacier. The user has access to numerous intrinsic parameters, and additional weighting parameters. Those are given in the table below.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyGr4jNode
Area	Compulsory	The area of the catchment represented by this node [km2]. E.g. Area = 123.4
Rainfall	Compulsory	An expression specifying the rainfall for the model. This can refer to a single dataset (e.g. Rainfall = MyRainData) or a linear combination of datasets (e.g. Rainfall = 0.6*MyRainDataA + 0.4*MyRainDataB). Rainfall data should be in mm.

Arguments	Arg Type	Value
Evaporation	Compulsory	Name of dataset containing evaporation data [mm]. E.g. Evaporation = MyEvaporationData
Tmin	Compulsory	Minimum daily temperature timeseries [Celcius]
Tmax	Compulsory	Maximum daily temperature timeseries [Celcius]
Elevation	Optional	Give the elevation of the node, and the reference elevation, in meters. If the reference elevation is not given it will be assumed to be 0m. E.g. elevation = 4500, 500
Lapse	Optional	Four values are defined in this line: tminlapse, tmaxlapse, petlapse, rainscal. Tminlapse and tmaxlapse describe how the min and max temperatures change with elevation [degrees/m]. Petlapse describes how the potential evapotranspiration decreases with elevation [mm/m]. Rainscal describes how rainfall increases with elevation [mm/km] (NOTE THE DIFFERENT UNITS FOR Rainscal AND Petlapse!). E.g. lapse = -0.0065, -0.0065, -0.0005, 1.0
Initial	Optional	Two values are defined in this line: InitialIce and InitialSnow. These are the initial depths (in mm) of the ice and snow buckets in the model. Defaults are 0. E.g. Initial = 0, 0
Parameters	Compulsory	There are 9 parameters defined here: First are the four GR4J parameters: X1, X2, X3, X4. Then we have 5 parameters associated with the Snow + Glacier component: ddfi, ddfs, tfrac, tmelt, taccum. Ddfi and Ddfs are the icemelt and snowmelt rates [mm/degree Celcius], tfrac is a number ranging from 0 to 1 defining how the representative temperature is calculated from the daily min and max. Tmelt and Taccum are temperature threshold for snowmelt and snow accumulation [Celcius]. E.g. parameters = 150, -3, 210, 1.5, 6, 4, 0.5, 0, 0
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode
SnowCoveredThreshold	Not exposed	Internal value is 10mm. This parameter relates to how snow cover is reported.
ThresholdSnowDepth	Not exposed	Internal value is 2.5mm. This is the snowbucket depth when ice melt processes become active (mm), default = 2.5, range = [0.001, 100]

Recorders:

OutNo	Description
0	Total node downstream flow [ML/day]
1	Runoff volume [ML/day]
2	Rainfall [mm/day]
3	Evaporation [mm/day]
4	Production Store [mm]
5	Routing Store [mm]
6	Snow area [m^2]

Example:

```
[GR4JSG]
name = mygr4jsgnode
area = 3821.2
rainfall = Rain
evaporation = evap
tmin = mintempdata
tmax = maxtempdata
elevation = 4500, 500
lapse = -0.0065, -0.0065, -0.0005, 1.0
initial = 0, 0
parameters = 150, -3, 210, 1.5, 6, 4, 0.5, 0, 0
dsnode = out
```

[GRSGDPIE]

A variant of GR4JSG developed by DPIE for Australian snow-capped catchments.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyGr4jNode
Area	Compulsory	The area of the catchment represented by this node [km2]. E.g. Area = 123.4
Rainfall	Compulsory	An expression specifying the rainfall for the model. This can refer to a single dataset (e.g. Rainfall = MyRainData) or a linear combination of datasets (e.g. Rainfall = 0.6*MyRainDataA + 0.4*MyRainDataB). Rainfall data should be in mm.
Evaporation	Compulsory	Name of dataset containing evaporation data [mm]. E.g. Evaporation = MyEvaporationData

Arguments	Arg Type	Value
Tmin	Compulsory	Minimum daily temperature timeseries [Celcius]
Tmax	Compulsory	Maximum daily temperature timeseries [Celcius]
Elevation	Optional	Give the elevation of the node, and the reference elevation, in meters. If the reference elevation is not given it will be assumed to be 0m. E.g. elevation = 4500, 500
Initial	Optional	Two values are defined in this line: InitialIce and InitialSnow. These are the initial depths (in mm) of the ice and snow buckets in the model. Defaults are 0. E.g. Initial = 0, 0

Parameters	Compulsory	There are 9 parameters here:
		
		parameters = 333, 0, 40, 0.5, + //x1, x2, x3, x4
		0.5, 0, 3.38, 1.3, 3 //tfrac, taccum, mrainfall, baserainfall, mnonrainfall
		
		x1, x2, x3, and x4 are as per GR4J.
		Tfrac is used to calculate a representative daily temperature relative to Tmin and Tmax. Tfrac~0 (Tfrac~1) means the temp behaves like Tmin (Tmax).
		Taccum the temperature [Celcius] when snow starts accumulating.
		Mrainfall is the rainfall melt rate
		Baserainfall is the rainfall base melt rate
		Mnorainfall the no-rainfall melt rate

Misc	Optional	There are 10 parameters here:
		
		misc = 0, 0, -0.0005, 0, + //tminlapseperm, tmaxlapseperm, petlapseperm, fluxaccumulation
		1, 6, 0, 0, 10, 0.5 //rainfallscalingfactor, ddfi, tmelt, percentaccumulation, snowcoveredthreshold, returnFlow
		
		Tminlapseperm is the lapse rate for the minimum daily temperature [C/m].
		Tmaxlapseperm is the lapse rate for the maximum daily temperature [C/m].
		Petlapseperm is the lapse rate for potential evapotranspiration [mm/m].
		Rainfallscalingfactor is a redundant rainfall scaling factor (ADDED FOR COMPLETELNESS BUT DONT USE IT; USE WEIGHTS IN THE RAINFALL DECLARATION INSTEAD).
		Ddfi is a relates ice melt rate to the temperature [mm/C].
		Tmelt is the temperature at which ice melts.
		Percentaccumulation is the percentage of area [%] which accumulates snow from snowfalls above.
		Snowcoverthreshold effects reporting only, and is the minimum snow depth [mm] that will be reported as snowcover.
		Returnflow is the time parameter [days] for refreeze unit hydrograph.

DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

Recorders:

OutNo	Description
0	Total node downstream flow [ML/day]
1	Runoff volume [ML/day]
2	Rainfall [mm/day]
3	Evaporation [mm/day]
4	Production Store [mm]
5	Routing Store [mm]
6	SnowArea [km2]
7	Total_Outflow_Depth
8	ModifiedRain
9	SnowMeltDepth
10	IceMeltDepth

Example:

```
[grsgdpie]
name = sc2_fu2
location = -119.7096, -555.1301
area = 68.115
rainfall = 1*rain_timeseries
evaporation = evap_timeseries
tmin = tmin_timeseries
tmax = tmax_timeseries
elevation = 1, 1 //catchment elevation, reference elevation
parameters = 350, 0, 40, 0.5, + //x1, x2, x3, x4
0.5, 0, 3.38, 1.3, 3 //tfrac, taccum, mrainfall, baserainfall, mnonrainfall
misc = 0, 0, -0.0005, 0, + //tminlapseperm, tmaxlapseperm, petlapseperm, fluxaccumulation
1, 6, 0, 0, 10, 0.5 //rainfallscalingfactor, ddfi, tmelt, percentaccumulation, snowcoveredthreshold, returnFlow
dsnode = sc2
```

[AWBM]

AWBM rainfall-runoff modelling functionality. There are 8 intrinsic AWBM parameters, and additional rainfall weighting parameters.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MyAwbmNode
Area	Compulsory	The area of the catchment represented by this node [km2]. E.g. Area = 123.4
Rainfall	Compulsory	An expression specifying the rainfall for the model. This can refer to a single dataset (e.g. Rainfall = MyRainData) or a linear combination of datasets (e.g. Rainfall = 0.6*MyRainDataA + 0.4*MyRainDataB). Rainfall data should be in mm.
Evaporation	Compulsory	Name of dataset containing evaporation data [mm]. E.g. Evaporation = MyEvaporationData
Parameters	Compulsory	The eight AWBM parameters: a1, a1, cap2, cap2, cap3, bfi, kbase, ksurf
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

Recorders:

OutNo	Description
0	Total node downstream flow [ML/day]
1	Runoff volume [ML/day]
2	Rainfall [mm/day]
3	Evaporation [mm/day]

Example:

```
[awbm]
name = Node 1
area = 1000.0
rainfall = rain_timeseries
evaporation = evap_timeseries
parameters = 0.134, 0.433, + //a1, a2
   7, 70, 150, + //cap1, cap2, cap3
   0.35, 0.95, 0.35 //bfi, kbase, ksurf
```

[Sacramento]

Fors provides Sacramento rainfall-runoff modelling functionality. There are two parts to this recipe: (1) First a Sacramento node should be defined using the "[sacramento]" tag and connected to upstream and downstream nodes as required. The Sacramento node is described in this section of the documentation. (2)Later a Sacramento model should be defined this is described later) and pointed to from your node. This arrangement allows several Sacramento nodes to use a common Sacramento model.

Arguments	Arg Type	Value
Name	Compulsory	Name of the node. This must be unique. E.g. Name = MySacramentoNode
Area	Compulsory	The area of the catchment represented by this node [km2]. E.g. Area = 123.4
Model	Compulsory	The name of the Sacramento model to attach this node to. E.g. Model = SacModel1
DSNode	Optional	Name of the downstream node. If there are multiple DSNode definitions, the node outflows will be divided equally amongst all downstream nodes. E.g. DSNode = MyDownstreamNode

OutNo	Description	Comment
0	Total node downstream flow [ML/day]	
1	Runoff volume [ML/day]	
2	Rainfall [mm/day]	
3	Evaporation [mm/day]	
4	Quickflow [ML/day]	
5	Slowflow [ML/day]	
6*	Quickflow_inter [ML/day]	

OutNo	Description	Comment
7*	Quickflow_surface [ML/day]	
8*	IQQM flosf [ML/day]	
9*	IQQM floin [ML/day]	
10*	IQQM flobf [ML/day]	
11*	IQQM flwsf [ML/day]	
12*	IQQM flwbf [ML/day]	
13*	IQQM roimp [ML/day]	
14	Uzfwc	Added in version n189
15	Uztwc	Added in version n189
16	Lzfpc	Added in version n189
17	Lzfsc	Added in version n189
18	Lztwc	Added in version n189

* **Warning**: Prior to v195 output numbers 9-13 did not exist, and output numbers 6, 7, 8 used to return different results (IQQM flosf, IQQM floin, IQQM flobf respectively).

Example:

```
[sacramento]
name = mysac
model = mysacmodel
area = 1019
```

[Sacramento model]

The most handy Sacramento model diagram I've ever seen is this one

Arguments	Arg Type	Value
Name	Compulsory	Name of this model. This must be unique. E.g. Name = MySacramentoModel
Rainfall	Compulsory	An expression specifying the rainfall for the Sacramento model. This can refer to a single dataset (e.g. Rainfall = MyRainData) or a linear combination of datasets (e.g. Rainfall = 0.6*MyRainDataA + 0.4*MyRainDataB). Rainfall data should be in mm.
RainfallMean	Optional	A value that can be used to force the mean daily rainfall over the model simulation period [mm/d]. If provided by the user, this is value is used to renormalize the rainfall sequence AFTER it has been calculated with the rainfall weights. This optional argument was added to help solve issues to do with excessive freedom in the rainfall calibration. At this point I only anticipate using it in a minority of cases. E.g. RainfallMean = 1.94
Evaporation	Compulsory	Name of dataset containing evaporation data [mm]. E.g. Evaporation = MyEvaporationData
InitialMoistureFraction	Optional	Optional value for adding some initial soil moisture. Default value is 0 (empty). Largest value you probably want to use is 1 (full). For a given value, F, the soil storages are initialized like this: Uzfwc=Uzfwm*value, Uztwc=Uztwm*F, Lztwc=Lztwm*F, Lzfsc Lzfsm*F, Lzfpc=Lzfpm*F.
Parameters	Compulsory	There are sixteen parameters. (1) Adimp, (2) Lzfpm, (3) Lzfsm, (4) Lzpk, (5) Lzsk, (6) Lztwm, (7) Pctim, (8) Pfree, (9) Rexp, (10) Sarva, (11) Side, (12) SSout, (13) Uzfwm, (14) Uzk, (15) Uztwm, (16) Zperc. E.g. Parameters = 0, 45, 60, 0.01, 0.01, 150, 0, 0.11, 1.5, 0, 0.2, 0.01, 25, 0.2, 47, 15
UH/LagUH	Compulsory	Either “UH” or “LagUH” must be specified. UH is the unit hydrograph represented as a sequence of real numbers. These should add to 1. Any number of elements can be specified. E.g. UH = 0.78, 0.20, 0.02, 0.0. LagUH is an alternative representation of the unit hydrograph, capable of representing any two-day unit hydrograph (which may be preceded by any number of zeros). The LagUH is mainly intended for calibration. E.g. LagUH = 0.1.

You cannot record anything directly from the Sacramento model, but you can record many results from the Sacramento node(s) pointing to this model.

Example:

```
[sacramento model]
name = mysacmodel
rainfall = rain
evaporation = evap
parameters = 0, 45, 60, 0.01, +
   0.01, 150, 0, 0.11, +
   1.5, 0, 0.2, 0.01, +
   25, 0.2, 47, 15
UH = 0.9, 0.1
```

[Recorder]

A recorder is used to record the results from a fors model run and output them to a csv file. A recorder is set using the ”[recorder]” tag.

Arguments	Arg Type	Value
Node	Compulsory	Name of the node whose results you want to record. E.g. Node = myNode
Outno	Compulsory	Number of the model result you want to record. The available results vary depending on node type. They are listed (with their corresponding outno values) in the descriptions each node type. E.g. outno = 1
Filename	Optional	Name of output file to save the result timeseries to. E.g. Filename = ModelledFlow.csv

Example:

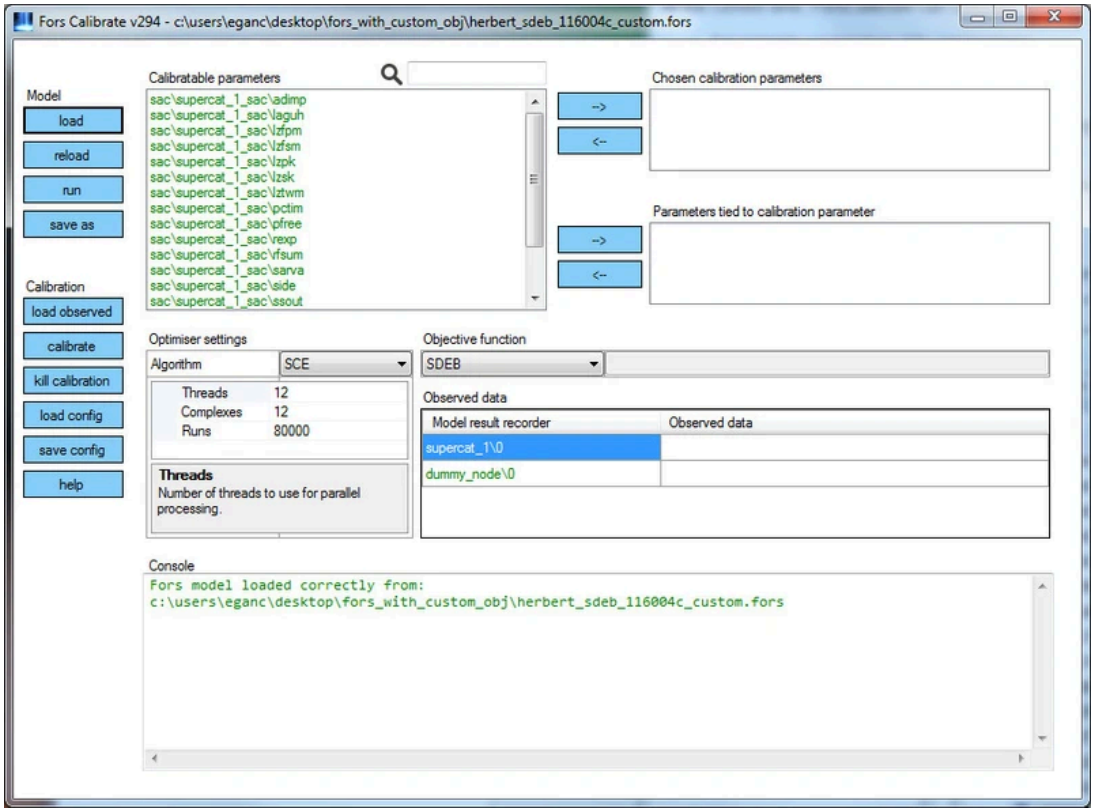
```
[recorder]
node = mysacnode
outno = 1
filename = modelledresult.csv
```

Calibrating a fors model

The ForsCalibrate tool works to find model parameter values that minimize the difference, as measured by the “objective function”, between the modeled results and the target timeseries. Several in-built objective functions are available, as well as the option for a custom objective function entered as an expression. Several optimization algorithms are available including SCE and CMAES.

How to best use ForsCalibrate depends on your situations, but a basic workflow may go something like this:

1. Prepare your fors model in a text editor
2. Load the fors model in the ForsCalibrate tool
3. Load some target timeseries data (e.g. observed flow data)
4. Choose the parameters you want to calibrate (e.g. all Sacramento parameters)
5. Choose a suitable objective function (e.g. sdeb)
6. Start the calibration by pressing the “calibrate” button
7. After the calibration finishes, the calibrated model result timeseries will be output automatically. Use Source (or other graphing/stats tool) to investigate the model results.
8. If you wish to save the Fors model, including the final calibrated parameter values, click the “save as” button.



Target timeseries data

You may calibrate any model result that your fors model is recording. Typically you would calibrate the modelled flow, e.g. at a streamflow station where you have some historic observed streamflow data. To set the target timeseries data with a particular model result, find the model result you want to calibrate under “model result recorder”, click on it to select it, then click “load observed”.

It is possible to assign target timeseries data to multiple recorders if you wish. This makes it possible to calibrate to multiple targets at once. The way ForsCalibrate balances model performance over multiple targets is described in the section on Objective Functions below.

Calibratable parameters

At the current time, ForsCalibrate can calibrate:

- Sacramento parameters (19+ parameters)
- GR4J parameters (5+ parameters)
- Nonlinear routing parameters (2 parameters)

Choose the parameters you want to calibrate by selecting them in the **“Calibratable parameters”** list and using the arrow button to move them to the **“Chosen calibration parameters”** list. Any parameters not moved will not be calibrate, and will retain their original values.

Ranges for the parameters are hard coded (if properly incentivised, Chas will change this).

Sacramento parameter ranges

Allowed ranges for the Sacramento parameters were derived in consideration of recommendations from Burnash, Weeks, and a review of previous calibrations undertaken at Queensland Hydrology (..\waarchive\Archive_(Queensland_Hydrology)\A_Surface_Water\13_Qld Best Practise Model Development and QA Guidelines\Sacramento\Qld. hydrology Sac Parameters 2013.docx).

These and are inbuilt in Fors, and are as follows:

Parameter	Description	Lower bound	Upper bound
Adimp	Basin fraction that becomes impervious with rain	0.00001	0.15
Lzfpm	Lower zone free water primary storage maximum (mm)	1	300
Lzfsm	Lower zone free water supplementary storage maximum (mm)	1	350
Lzpk	Lower zone primary drainage rate	0.001*Lzsk	1.0*Lzsk
Lzsk	Lower zone supplementary drainage rate	0.001	0.9
Lztwm	Lower zone tension water storage maximum (mm)	10	600
Pctim	Permanent impervious fraction of catchment draining to channel	0.00001	0.11

Parameter	Description	Lower bound	Upper bound
Pfree	Proportion of percolated water transferred directly to the Lower Zone Free Water Storages	0.01	0.5
Rexp	Exponent in the percolation equation	1	6
Rserv	Fraction of Lower Zone Free Water storages unavailable for transpiration	0.3	0.3
Sarva	Fraction of the catchment covered by water and riparian vegetation	0.0001*Pctim	1.0*Pctim
Side	Channel loss ratio	0.00001	0.1
Ssout	Channel loss, subsurface channel flow (mm/day)	0.00001	0.1
Uzfwm	Upper zone free water storage maximum (mm)	5	155
Uzk	Upper zone lateral drainage rate	0.1	1
Uztwm	Upper zone tension water storage maximum (mm)	12	180
Zperc	Proportional increase in percolation from saturated to dry conditions	1	600
LagUH	Reparametrized unit hydrograph	0	3
RFSUM	The total rainfall weight, i.e. the sum of all rainfall factors (weights).	0.7	3
RFP?	Rainfall factor proportion constants. These are used to apportion the total rainfall weight amongst the individual stations. There are n-1 of these. They are used to determine the rainfall weights as follows (this example is for a system with 4 rainfall stations): Weight1 = Rfsum * Rfp0 Weight2 = (Rfsum - Weight1) * Rfp1 Weight3 = (Rfsum - Weight1 - Weight2) * Rfp2 Weight4 = (Rfsum - Weight1 - Weight3 - Weight4)	0	1

GR4J parameter ranges

Parameter	Description	Lower bound	Upper bound
x1	GR4J x1 parameter	10	2000
x2	GR4J x2 parameter	-8	6
x2	GR4J x3 parameter	10	500
x2	GR4J x4 parameter	0	4
RFSUM	Rainfall weighting factor	0.7	3
RFP?	Rainfall weighting proportion (1 for each station beyond the first)	0	1

Nonlinear routing parameter ranges

Parameter	Description	Lower bound	Upper bound
kperkm	Nonlinear routing storage constant per km	1	100,000
m	Nonlinear routing exponent	0.6	0.9

Objective functions

An objective function is a measure of the goodness-of fit between model results and target results. In choosing a particular objective function, you are prescribing a quantitative definition for the goodness-of-fit that the optimizer can use to decide whether one simulation is a better (or worse) fit than another simulation. The objective function implicitly determines how the optimizer balances the importance of low-flows, high-flows, various flow-statistics, and how it balances the value of performance at different sites (if calibrating to multiple sites).

To make life easy, there are several inbuilt objective functions. And to fulfill more advanced needs, custom expression can be used do tailor your own objective function. The inbuilt and custom objective functions are detailed below.

Inbuilt objective functions

The fors calibration tool has several in-built objective functions. If you are working with streamflow data, “SDEB” is probably a good choice.

Abbreviation	Description
SDEB	Targets square-root daily flows + exceedance, total bias. $\mathrm{SDEB} = \left[0.1\sum_i \left(\sqrt{Q_i} - \sqrt{q_i}\right)^2 + 0.9\sum_i \left(\sqrt{R_i} - \sqrt{r_i}\right)^2\right] \times \left(1 + \frac{\lvert\sum_i Q_i - \sum_i q_i\rvert}{\sum_i Q_i}\right)$
SDB	Targets square-root daily flows, and total bias. $\mathrm{SDB} = \left[\sum_i \left(\sqrt{Q_i} - \sqrt{q_i}\right)^2\right] \times \left(1 + \frac{\lvert\sum_i Q_i - \sum_i q_i\rvert}{\sum_i Q_i}\right)$
NSE	Targets Nash-Sutcliffe Efficiency coefficient. Compared to other objective functions, this tends to target larger flows and be relatively insensitive to lower flows. Since better performance is indicated by larger values of NSE, the optimizer actually works by minimizing -NSE. $\mathrm{NSE} = 1 - \frac{\sum_i (Q_i - q_i)^2}{\sum_i (Q_i - \bar{Q})^2}$
PEARS_R	Pearson's correlation coefficient R. Since better performance is indicated by larger values of pears_r, the optimizer actually works by minimizing -PEARS_R. $PEARS\_R = \frac{\sum_i (q_i - \overline{q})(Q_i - \overline{Q})}{\sqrt{\sum_i (q_i - \overline{q})^2}\sqrt{\sum_i (Q_i - \overline{Q})^2}}$

(Here Qi is the observed flow and qi is the modelled flow on day i.)

Custom objective function

If you choose CUSTOM from the **Objective function** drop-down, you will enable the adjacent text-box. Define your custom objective by typing an expression in this box. Expressions can be constructed using the following constants, mathematical functions, and timeseries functions. (Sorry for the bare-bones documentation. I hope I've given enough for you to work it out).

Constant	Value
pi	3.141…

Standard Mathematical and Logical Functions
+ - * / %
abs(?) sin(?) cos(?) tan(?) asin(?) acos(?) atan(?) pow(?,?) exp(?) exp10(?) log10(?) ln(?) logbase(?,?) and(?,?) or(?,?) not(?) xor(?,?) if(?,?,?) ceiling(?) floor(?) round(?) max(?,?,…) min(?,?,…) gt(?,?) lt(?,?) ge(?,?) le(?,?) eq(?,?) neq(?,?) rand()

Timeseries Function	Arguments	Description
bias(?)	1. index of the result-target pair	Calculates the bias (mod_tot/target_tot) over the common period
biaspenalty(?)	1. index of the result-target pair	Calculates a penalty term: 1 + abs((mod_tot - target_tot)/target_tot)
pears_r(?)	1. index of the result-target pair	Calculates the Pearson's R over the common period
nse(?)	1. index of the result-target pair	Calculates the Nash-Sutcliffe Efficiency over the common period
sdb(?)	1. index of the result-target pair	Calculates SDB over the common period
pdeb(?,?,?,?,?)	1. index of the result-target pair 2. lambda 3. alpha 4. nu 5. mu	Calculates PDEB over the common period. $\mathrm{PDEB} = \left[\alpha\sum_i (Q_i^\lambda - q_i^\lambda)^2 + (1-\alpha)\sum_i (R_i^\lambda - r_i^\lambda)^2\right]^\nu \times \left(1 + \frac{\lvert\sum_i Q_i - \sum_i q_i\rvert}{\sum_i Q_i}\right)^\mu$
ldeb(?,?,?,?,?)	1. index of the result-target pair 2. epsilon 3. alpha 4. nu 5. mu	Calculates LDEB over the common period. $\mathrm{LDEB} = \left[\alpha\sum_i (\log[Q_i+\epsilon] - \log[q_i+\epsilon])^2 + (1-\alpha)\sum_i (\log[R_i+\epsilon] - \log[r_i+\epsilon])^2\right]^\nu \times \left(1 + \frac{\lvert\sum_i Q_i - \sum_i q_i\rvert}{\sum_i Q_i}\right)^\mu$

Timeseries Function	Arguments	Description
sdeb(?)	1. index of the result-target pair	Calculates SDEB over the common period
ndata(?)	1. index of the result-target pair	The number of data points in the common period

Example custom function expressions

Example	Meaning
sdeb(0)	SDEB value for result-target pair 0
sdeb(0)/ndata(0)	SDEB value for result-target pair 0 divided by the number of days
15.0 * sdeb(0) + 9.2 * sdeb(1)	A weighted combination of the SDEB values at result-target pairs 0 and 1
pdeb(0,0.5,0.1,1,1) * biaspenalty(1)	The PDEB value of result-target pair 0, with root power (lambda=0.5), equal daily and exceedance balance (alpha=0.5), linear exponents (nu=1, mu=1), multiplied by a bias penalty term calculated on result-target pair 1 (which may compare, for example, the modeled and expected baseflow components).
sdeb(0) * (1 + abs((bias(1)/bias(0) - 0.6)/(0.6)))	Here is a cool example where I want to force the slow-flow fraction to be around 60%. Check out this example in more detail here.

Optimization algorithms

ForsCalibrate has a few optimization algorithms you can choose from.

In the basic case where you're just calibrating a single Sacramento model, the default SCE algorithm does a good job.

If you're doing many more parameters (e.g. 2 or more independent Sacramento models) then I think the CMAES algorithm may have some advantages. This suspicion is based on limited testing… so do some testing!

Algorithm	Description
SCE	Shuffled Complex Evolution (Duan 1991, 1992, 1994)
CMAES	Covariance Matrix Adaptation Evolution Strategy (Hansen 2004, 2006)
RB	Rosenbrock local search algorithm (CRAP - thwarted by local minima)
GRID	Basic grid search (CRAP - exponentially slow)

Running a fors model from the commandline

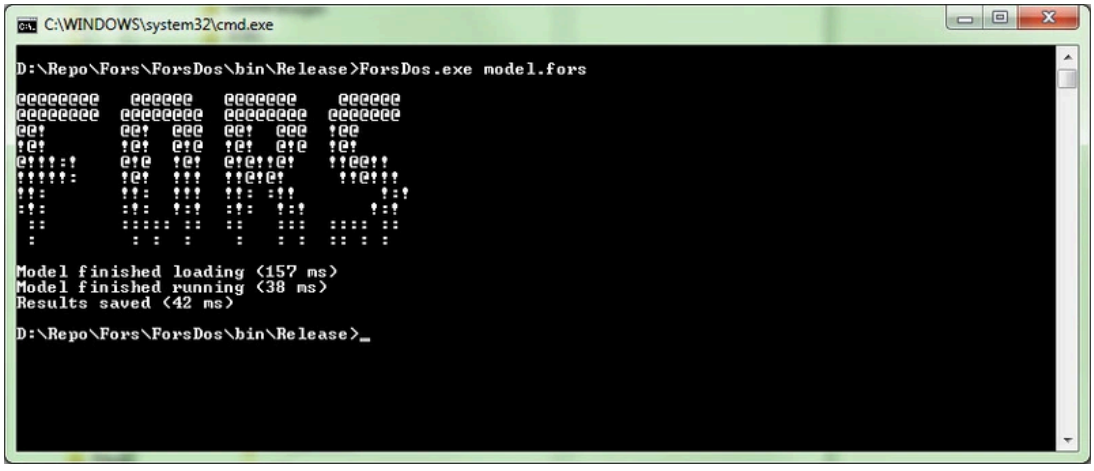
There is a commandline runner for fors called ForsDos.exe. This allows you to run a fors model, or run a fors calibration, from the commandline. This may be handy for batching fors model runs or fors calibration runs, or even driving fors using external tools such as PEST, RUNDMM, RUNSPER, etc.

ForsDos takes a single argument being the name of your model file, or the calibration settings file. Usage is as follows:

```
forsdos model.fors
```

or

```
forsdos mycalibration.calib
```



Using ForsGaugeDataPrep

See full documentation here.

When working with residual reaches you typically only want to calibrate during periods where all upstream and downstream gauges have good quality data. The program “ForsGaugeDataPrep.exe” can be used to strip data that is not in the common period.

ForsGaugeDataPrep should be run from the commandline like this:

```
> ForsGaugeDataPrep myInputFile.txt
```

Where the input file looks something like this:

```
422306.csv, 1, 422306_common.csv
422313.csv, 1, 422313_common.csv
422394.csv, 1, 422394_common.csv
422310.csv, 0, 422310_common.csv
```

The first column contains the original gauge data files. The second column contains relative lag values which can be used to account for lag between upstream and downstream gauges. The third column contains output file names.

Example 1: Headwater Sacramento

```
[input]
file = 9170050_e.csv
name = evap

[input]
file = 9170050_r.csv
name = rain

[sacramento]
name = mysacnode
model = mysacmodel
area = 218

[sacramento model]
name = mysacmodel
rainfall = rain
evaporation = evap
parameters = 0, 45, 60, 0.01, +
  0.01, 150, 0, 0.11, +
  1.5, 0, 0.2, 0.01, +
  25, 0.2, 47, 15
UH = 0.9, 0.1

[recorder]
node = mysacnode
outno = 0
filename = 917005_fors.csv
```

Example 2: Residual Reach

```
[input]
file = 422306.csv
name = 422306_full

[input]
file = 422310_common.csv
name = 422310_common

[input]
file = 422313.csv
name = 422313_full

[input]
file = 422394.csv
name = 422394_full

[input]
file = zeros.csv
name = zeros

[input]
file = 41013_pp_rain.csv
name = rain

[input]
file = 41013_pp_mpot_rolled.csv
name = evap

////////////////////////////////////// reach 1

[inflow]
name = 422306a_inflow
inflow = zeros
dsnode = 422306a_gauge

[gauge] //83
name = 422306a_gauge
```

```
observedflow = 422306_full
applyobserved = true
dsnode = confluence

////////////////////////////////////// reach 2

[inflow]
name = 422313ab_inflow
inflow = zeros
dsnode = 422313ab_gauge

[gauge] //at b 147
name = 422313ab_gauge
observedflow = 422313_full
applyobserved = true
dsnode = confluence

////////////////////////////////////// reach 3

[inflow]
name = 422394a_inflow
inflow = zeros
dsnode = 422394a_gauge

[gauge] //327
name = 422394a_gauge
observedflow = 422394_full
applyobserved = true
dsnode = confluence

////////////////////////////////////// reach 4

[node]
name = confluence
dsnode = my_routing

[nonlinear reach]
name = my_routing
parameters = 100000, 0.75
dsnode = residual_gauge

[gauge]
name = residual_gauge
observedflow = 422310_common
dsnode = mysac

[sacramento]
name = mysac
area = 913
model = mysacmodel
dsnode = 422310abc_gauge

[gauge]
name = 422310abc_gauge
observedflow = 422310_common

////////////////////////////////////// ALL DONE

[sacramento model]
name = mysacmodel
rainfall = rain
evaporation = evap
parameters = 0, 45, 60, 0.01, +
    0.01, 150, 0, 0.11, +
    1.5, 0, 0.2, 0.01, +
    25, 0.2, 47, 15
uh = 0.95, 0.05

[recorder]
node = 422310abc_gauge
outno = 0
filename = 422310abc_modelled.csv

[recorder]
node = mysac
outno = 1
filename = 422310abc_residual_sac.csv

[recorder]
node = residual_gauge
outno = 2
filename = 422310abc_calculated_residual.csv

[recorder]
node = residual_gauge
outno = 0
filename = 422310abc_routed_upstream.csv
```

Reference Material

Anderson (2002), NOAA 1_Anderson_CalbManual.pdf