Command Reference: AnalyzeNetworkPointFlow()

Analyze a node/link network to calculate "point flow" for nodes

Version 10.31.00, 2014-07-22

The `AnalyzeNetworkPointFlow()` command takes as input information to define a "flow network", associates input time series with each node in the network, and computes mass balance time series at each node. Although the network is intended to represent a physical network such as a stream system, it also can represent other flow networks such as transportation or other mass/energy conservation systems.

This command differs from the functionality of other network analysis tools as follows:

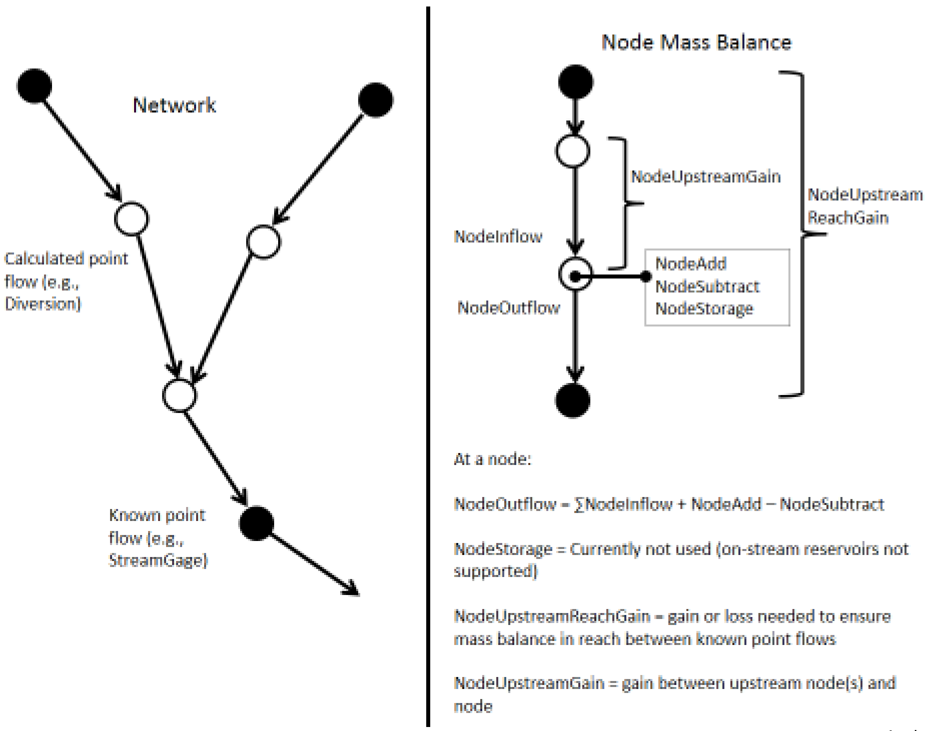
- Daily administration tools, such as the State of Colorado's Colorado Water Rights Administration Tool (CWRAT) perform a point flow analysis for a single day, which only requires knowing one day's input values, whereas `AnalyzeNetworkPointFlow()` analyzes time series for a specified period.
- More sophisticated models, such as the State of Colorado's StateMod water allocation model, perform allocation decisions within each time step for the full period, whereas `AnalyzeNetworkPointFlow()` performs a sequence of basic time series manipulations that can be quickly configured.

It may be possible to utilize the network data from tools such as those mentioned above with the `AnalyzeNetworkPointFlow()` command. For example, the StateMod river network file *(*.rin)* can be read using the `TSTool ReadTableFromFixedFormatFile()` command.

Limitations to be addressed with future enhancements:

- The command does not handle branching networks in calculations
- The command does not handle reservoirs and storage calculations
- The output time series identifiers and alias cannot be user-defined
- Temporary known flows, for example at a diversion that dries the river, are not handled in calculations

The following figure illustrates the network connectivity and mass balance that is performed at each node. Currently "on-channel" reservoirs with storage are not supported and gain/loss can only be computed in non-branching networks – these features and others necessary to model more complex networks will be added in the future; however, this command is not intended to replace more complex models. Consequently the command currently is suitable for analysis of a main stem river with no on-channel reservoirs.



AnalyzeNetworkPointFLow

AnalyzeNetworkPointFlow() Network and Node Mass Balance

There are two main data configuration requirements:

1. Define the node network using node identifiers, node type, and other information that control the analysis.
2. Provide data that allows the command to locate the input time series for the analysis by matching with the node. The time series identifiers (or alias) used by TSTool generally have location identifiers that match the node identifier; however, this is not a requirement if the input TSID is specified directly.

Defaults are in place that allow the time series identifier to be created from the network data. However, for complicated networks where input time series may be accessed from multiple sources, a general default may not result in matching time series. Two examples are shown below to illustrate the different approach for specifying time series.

The following example shows a network that is defined in Excel as a table containing a list of node identifiers with associated properties, as illustrated in the following figure. In this case, the node identifiers will be used by default to create the time series identifier to match input time series (because the `TSIDColumn` command parameter is not specified and there is no TSID column in the network table). In this case, the input time series TSIDs must match the node identifier and the data type provided with the `NodeAddDataTypes` and similar command parameters.

	A	B	C	D	E	F
1	NodeID	NodeName	NodeType	NodeDist	NodeWeight	DownstreamNodeID
2	06754000	*KERSEY GAGE*	StreamGage	2.0		0103816
3	0103816	EMPIRE RESERVOIR INLET (at reservoir)	Diversion	3.0	1.0	0100503
4	0100503	RIVERSIDE CANAL (at reservoir)	Diversion	4.0	1.0	0100507
5	0100507	BIJOU CANAL	Diversion	5.0	1.0	0100513
6	0100513	JACKSON LAKE INLET	Diversion	6.0	1.0	0100511
7	0100511	WELDON VALLEY DITCH	Diversion	7.0	2.0	0100512
8	0100512	JACKSON LAKE OUTLET DITCH	Return	8.0	2.0	0100514
9	0100514	FT MORGAN CANAL	Diversion	9.0	3.0	Instream1
10	Instream1	Instream Flow 1	InstreamFlow	10.0	3.0	06758500
11	06758500	*WELDONA GAGE*	StreamGage	11.0	3.0	0102900
12	0102900	WELDON VALLEY RETURN	Return	13.0	3.0	0100517
13	0100517	DEUEL AND SNYDER	Diversion	15.0	4.0	0100515
14	0100515	UPPER PLATTE BEAVER CNL	Diversion	17.0	4.0	06759500
15	06759500	*FORT MORGAN GAGE*	StreamGage	19.0	4.0	0100518
16	0100518	LOWER PLATTE BEAVER D	Diversion	26.0	4.0	0100519
17	0100519	TREMONT DITCH	Diversion	37.0	5.0	0100687
18	0100687	NORTH STERLING CANAL	Diversion	38.0	5.0	0100688
19	0100688	UNION DITCH	Diversion	49.0	5.0	06760000
20	06760000	*BALZAC GAGE*	StreamGage	50.0	5.0	

AnalyzeNetworkPointFlow_InputTable

AnalyzeNetworkPointFlow() Network Input Table – Input Time Series Matched with NodeID

The following example illustrates how the input time series identifier can be specified in the network data. The TSID column in the table in this case contains time series aliases that will be matched with input time series, where %I will be replaced with the analysis interval specified by the `Interval` command parameter. Using an alias for input time series allows a common naming convention for time series to be implemented, which can simplify output (the period-delimited time series identifier is still used with the time series but the alias takes precedence).

	A	B	C	D	E	F	G	H	I
1	C	NodeID	NodeName		NodeType	NodeDist	DataType	DownstreamNodeID	TSID
2		06752000	*CACHE LA POUDRE RIV AT M*		StreamGage	**0.00000**	Streamflow	0300908	06752000-Streamflow-%I
3		0300908	GREELEY FILTERS PIPELINE		Diversion	**0.40000**	DivTotal	HSCPVDCO	0300908-Diversion-%I
4		HSCPVDCO	HANSEN SUP CAN TO POUDI		Return	**1.01001**	Streamflow	0300910	HSCPVDCO-Return-%I
5		0300910	PLEASANT VALLEY LAKE CNL		Diversion	**1.29000**	DivTotal	0300911	0300910-Diversion-%I
6		0300911	LARIMER COUNTY DITCH		Diversion	**2.38000**	DivTotal	0300912	0300911-Diversion-%I
7		0300912	DRY CREEK DITCH		Diversion	4.46000	DivTotal	0300913	0300912-Diversion-%I
8		0300913	NEW MERCER @ 10 FOOT PA		Diversion	**5.70001**	DivTotal	0300914	0300913-Diversion-%I
9		0300914	LARIMER COUNTY #2 @ 12 FT		Diversion	**5.70002**	DivTotal	0300915	0300914-Diversion-%I
10		0300915	CACHE LA POUDRE DITCH		Diversion	**5.70003**	DivTotal	0301029	0300915-Diversion-%I
11		0301029	Taylor & Gill		Diversion	**5.70004**	DivTotal	0303780	0301029-Diversion-%I
12		0303780	Claymore Outlet		Return	**5.74000**	RelTotal	0300918	0303780-Return-%I
13		0300918	FORT COLLINS IRR CNL		Diversion	**7.99000**	DivTotal	0300919	0300918-Diversion-%I
14		0300919	LARIMER AND WELD IRRIGAT		Diversion	**8.33000**	DivTotal	0300922	0300919-Diversion-%I
15		0300922	LAKE CANAL DITCH		Diversion	**10.32000**	DivTotal	0300923	0300922-Diversion-%I
16		0300923	JOHN G COY DITCH		Diversion	**10.62000**	DivTotal	06752260	0300923-Diversion-%I
17		06752260	*CACHE LA POUDRE RIVER AT*		StreamGage	**11.46000**	Streamflow	FtCollinsWWTP1	06752260-Streamflow-%I

AnalyzeNetworkPointFlow_InputTable_TSID

AnalyzeNetworkPointFlow() Network Input Table – Input Time Series Specified with TSID Column

In the above examples the network is defined in an Excel file, the `ReadTableFromExcel()` command is used to read the table, and the table is used as input to the `AnalyzeNetworkPointFlow()` command

The network definition table columns from the above figure are as follows (note, however, that the column names are user-defined and are specified as parameters to the `AnalyzeNetworkPointFlow()` command:

AnalyzeNetworkPointFlow() Network Input Table Column Description

Example Network Table Column	Command Parameter to Indicate Column	Description
`NodeID`	`NodeIDColumn`	The location ID for the network node, typically corresponding to the location ID in time series identifiers.
`NodeName`	`NodeNameColumn`	The node name, useful because `NodeID` is generally terse and non-descriptive, used in messages.
`NodeType`	`NodeTypeColumn`	The node type, needed to define node behavior (e.g., whether time series values get added, subtracted, reset at node). The node types are user-defined, although types often are defined by modeling conventions. The behavior corresponding to node types is defined by using command parameters (`NodeAddTypes`, `NodeSubtractTypes`, `NodeOutflowTypes`, `NodeFlowThroughTypes`).
`NodeDist`	`NodeDistanceColumn`	The node distance along the flow path. Typically the distance is measured relative to the lowest point on the network. The distance is used to estimate gain/loss when `GainMethod=Distance` is specified as a command parameter.
`NodeWeight`	`NodeWeightColumn`	Used when `GainMethod=Weight`. The weights indicate the relative weight of the reach gain/loss to be distributed between nodes on the reach. For example, specify a best estimate of the percentage of reach loss that occurs above each node. Or, specify as a rate of gain/loss when used with `GainMethod=DistanceWeight` (but in this case the distance*weight product will be normalized to ensure that the reach gain/loss is equalized between known point flows).
`Downstream NodeID`	`DownstreamNodeID Column`	The location ID for the downstream node, needed to define network connectivity.
`TSID`	`TSIDColumn`	Indicates the time series identifier or alias for the input time series. The notation `%I` can be used as a place holder that will be replaced with value of the Interval command parameter. In the future, additional similar columns will be added if multiple input time series are needed with other node types.

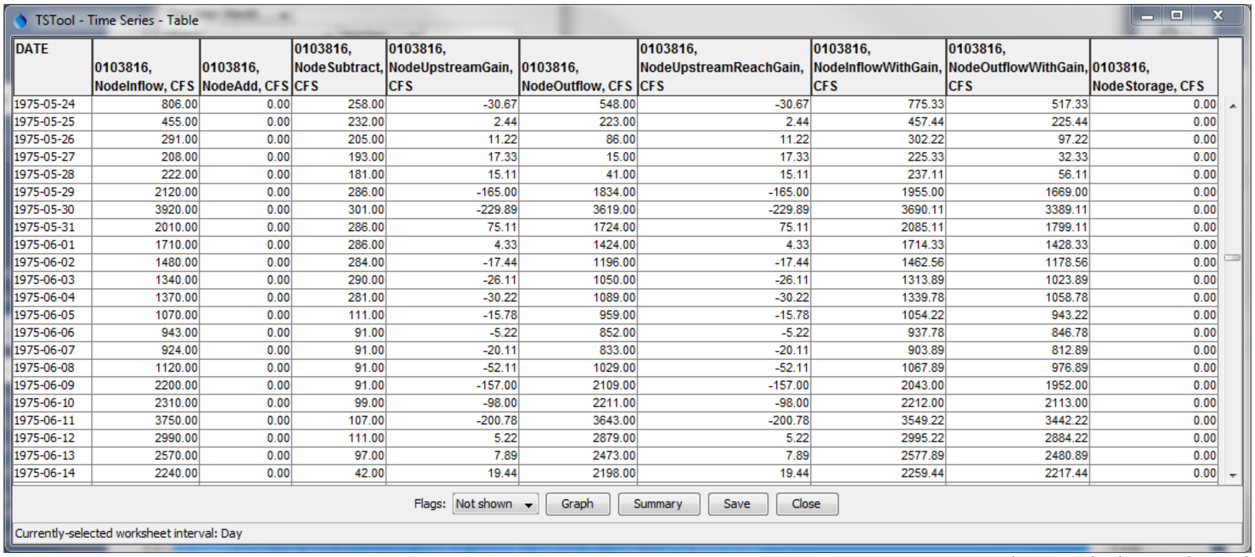
The `AnalyzeNetworkPointFlow()` command creates output time series with the data types indicated in the following table. For example, the time series identifier for the output time series will be similar to the following (where the `NodeID` is taken from the network, data source is not used, data type is as listed below, and the interval agrees with the `Interval` command parameter):

```
NodeID..NodeInflow.Day
```

AnalyzeNetworkPointFlow() Network Input Table Column Description

Column	Description
`NodeInflow`	Sum of outflows from upstream nodes, which are consequently inflows to the current node (lagged routing currently is not implemented).
`NodeAdd`	Time series added at the node (for example immediately off-channel reservoir release or measured return flow).
`NodeSubtract`	Time series subtracted at the node (for example diversion).
`NodeUpstreamGain`	Gain (positive) or loss (negative) between immediate upstream node(s) and the current node (missing if gain/loss is not computed).
`NodeOutflow`	Outflow from the node, which takes into account inflow and any additions and subtractions at the node. This is essentially natural flow.
`NodeUpstreamReachGain`	Gain (positive) or loss (negative) between upstream known flow node(s) and the current node (missing if gain/loss is not computed).
`NodeInflowWithGain`	`NodeInflow` + `NodeUpstreamReachGain` (missing if gain/loss are not computed).
`NodeOutflowWithGain`	`NodeOutflow` + `NodeUpstreamReachGain` (missing if gain/loss are not computed).
`NodeStorage`	Storage at the node after additions and subtractions (currently always zero, will enhance in the future to handle on-channel reservoirs).

The following figure illustrates the output time series corresponding to the data types listed in the above table:



TSTool - Time Series - Table

DATE	0103816, NodeInflow, CFS	0103816, NodeAdd, CFS	0103816, NodeSubtract, CFS	0103816, NodeUpstreamGain, CFS	0103816, NodeOutflow, CFS	0103816, NodeUpstreamReachGain, CFS	0103816, NodeInflowWithGain, CFS	0103816, NodeOutflowWithGain, CFS	0103816, NodeStorage, CFS
1975-05-24	806.00	0.00	258.00	-30.67	548.00	-30.67	775.33	517.33	0.00
1975-05-25	455.00	0.00	232.00	2.44	223.00	2.44	457.44	225.44	0.00
1975-05-26	291.00	0.00	205.00	11.22	86.00	11.22	302.22	97.22	0.00
1975-05-27	208.00	0.00	193.00	17.33	15.00	17.33	225.33	32.33	0.00
1975-05-28	222.00	0.00	181.00	15.11	41.00	15.11	237.11	56.11	0.00
1975-05-29	2120.00	0.00	286.00	-165.00	1834.00	-165.00	1955.00	1669.00	0.00
1975-05-30	3920.00	0.00	301.00	-229.89	3619.00	-229.89	3690.11	3389.11	0.00
1975-05-31	2010.00	0.00	286.00	75.11	1724.00	75.11	2085.11	1799.11	0.00
1975-06-01	1710.00	0.00	286.00	4.33	1424.00	4.33	1714.33	1428.33	0.00
1975-06-02	1480.00	0.00	284.00	-17.44	1196.00	-17.44	1462.56	1178.56	0.00
1975-06-03	1340.00	0.00	290.00	-26.11	1050.00	-26.11	1313.89	1023.89	0.00
1975-06-04	1370.00	0.00	281.00	-30.22	1089.00	-30.22	1339.78	1058.78	0.00
1975-06-05	1070.00	0.00	111.00	-15.78	959.00	-15.78	1054.22	943.22	0.00
1975-06-06	943.00	0.00	91.00	-5.22	852.00	-5.22	937.78	846.78	0.00
1975-06-07	924.00	0.00	91.00	-20.11	833.00	-20.11	903.89	812.89	0.00
1975-06-08	1120.00	0.00	91.00	-52.11	1029.00	-52.11	1067.89	976.89	0.00
1975-06-09	2200.00	0.00	91.00	-157.00	2109.00	-157.00	2043.00	1952.00	0.00
1975-06-10	2310.00	0.00	99.00	-98.00	2211.00	-98.00	2212.00	2113.00	0.00
1975-06-11	3750.00	0.00	107.00	-200.78	3643.00	-200.78	3549.22	3442.22	0.00
1975-06-12	2990.00	0.00	111.00	5.22	2879.00	5.22	2995.22	2884.22	0.00
1975-06-13	2570.00	0.00	97.00	7.89	2473.00	7.89	2577.89	2480.89	0.00
1975-06-14	2240.00	0.00	42.00	19.44	2198.00	19.44	2259.44	2217.44	0.00

Flags: Not shown | Graph | Summary | Save | Close

Currently-selected worksheet interval: Day

AnalyzeNetworkPointFLow_OutputTS

AnalyzeNetworkPointFlow() Output Time Series Table

The following logic is used to analyze the network. Currently this logic is performed by navigating the network from most upstream to downstream node and processing all timesteps for a node before moving to the next node. In the future the entire network may be traversed for each timestep to allow for temporary known flows within a reach.

1. The network is navigated from top to bottom. When a confluence is found (a node with more than one upstream node), each confluence is processed from the top down to the confluence point. Of particular importance is the concept of a "stream reach", which is the reach between known flow points, because mass balance is enforced at known flow points and gain/loss can be estimated between the known flow points.
    a. The data type for the node (see `*DataType` command parameters) is used to retrieve the relevant time series for the node. The first time series that matches the location ID, data type, and interval is used as input for the node. The time series must have been read prior to the `AnalyzeNetworkPointFlow()` command. For example, use the `CopyTable()` command to copy a subset of the network table's `NodeID` values and then use the `ReadTimeSeriesList()` command with the list of identifiers.
    b. Calculate the node's inflow:
        i. Node types that set outflow, indicated by the `NodeOutflowDataTypes` parameter (e.g., `StreamGage`):
            - `NodeInflow` = input time series for node
        ii. All other node types:
            - `NodeInflow` = sum of upstream node outflows
    c. Calculate the node's outflow:
        i. Node types that add, indicated by the `NodeAddDataTypes` parameter (e.g., `Return`, `Import`):
            - `NodeOutflow` = `NodeInflow` + added time series
        ii. Node types that subtract, indicated by the `NodeSubtractDataTypes` parameter (e.g., `Diversion`):

- `NodeOutflow` = `NodeInflow` - subtracted time series

iii. Node types that set outflow, indicated by the `NodeOutflowDataTypes` parameter (e.g., `StreamGage`):
- `NodeOutflow` = `NodeInflow`

iv. Node types that let flow through, indicated by the `NodeFlowThroughDataTypes` parameter (e.g., `InstreamFlow`):
- `NodeOutflow` = `NodeInflow`.

d. For known flow points (e.g., `StreamGage` node type), set the reach gain/loss:

i. `NodeUpstreamReachGain` = difference between upstream node outflow and known flow at downstream node in reach

e. If gain/loss is being estimated and a known flow node encountered (e.g., `StreamGage`), gain/loss between this node and the nearest upstream node(s) is compute. **This has only been implemented for the case where all intervening nodes are in a non-branching reach**.

i. First calculate the distribution factor by which the reach gain (see previous step) will be distributed to each node in the reach:
- If the `GainMethod=None`, no adjustment to flows is made and the gain/loss upstream of the know flow node will result in a discontinuous jump because no gain/loss adjustment is made.
- If the `GainMethod=Distance`, use the node distance data from the network table to prorate the gain/loss in the stream reach. The difference in distance between the upstream node and the current node is set to weight for prorating the reach gain/loss. Use this method if the gain/loss rate is the same throughout the reach and therefore only the distance between nodes controls the gain/loss.
- If the `GainMethod=Weight`, the gain/loss is prorated by the weights specified by the `NodeWeightColumn` parameter (or weight equally if the weights are not specified in the network table). The weight of the upstream known flow node is not used. Use this method if the relative gain/loss for each node within the reach can be specified.
- If the `GainMethod=DistanceWeight`, the gain/loss is prorated by the product of the weights specified by the `NodeWeightColumn` parameter (or weight equally if the weights are not specified in the network table) and by the values from the `NodeDistanceColumn`. The weight of the upstream known flow node is not used. Use this method if the relative rate of gain/loss for each node can be specified, but overall gain/loss is also a function of the distance. Even though a rate is specified, the calculated gain/loss may be slightly different because the overall reach gain/loss must be balanced at known flow points for each time step.

- Multiply `NodeUpstreamReachGain` by the gain/loss distribution factor to calculate `NodeReadGain` for each node.
- Compute the cumulative gain/loss for the node by summing `NodeUpstreamNodeGain` for each upstream node and set to `NodeUpstreamReachGain` for the current node.

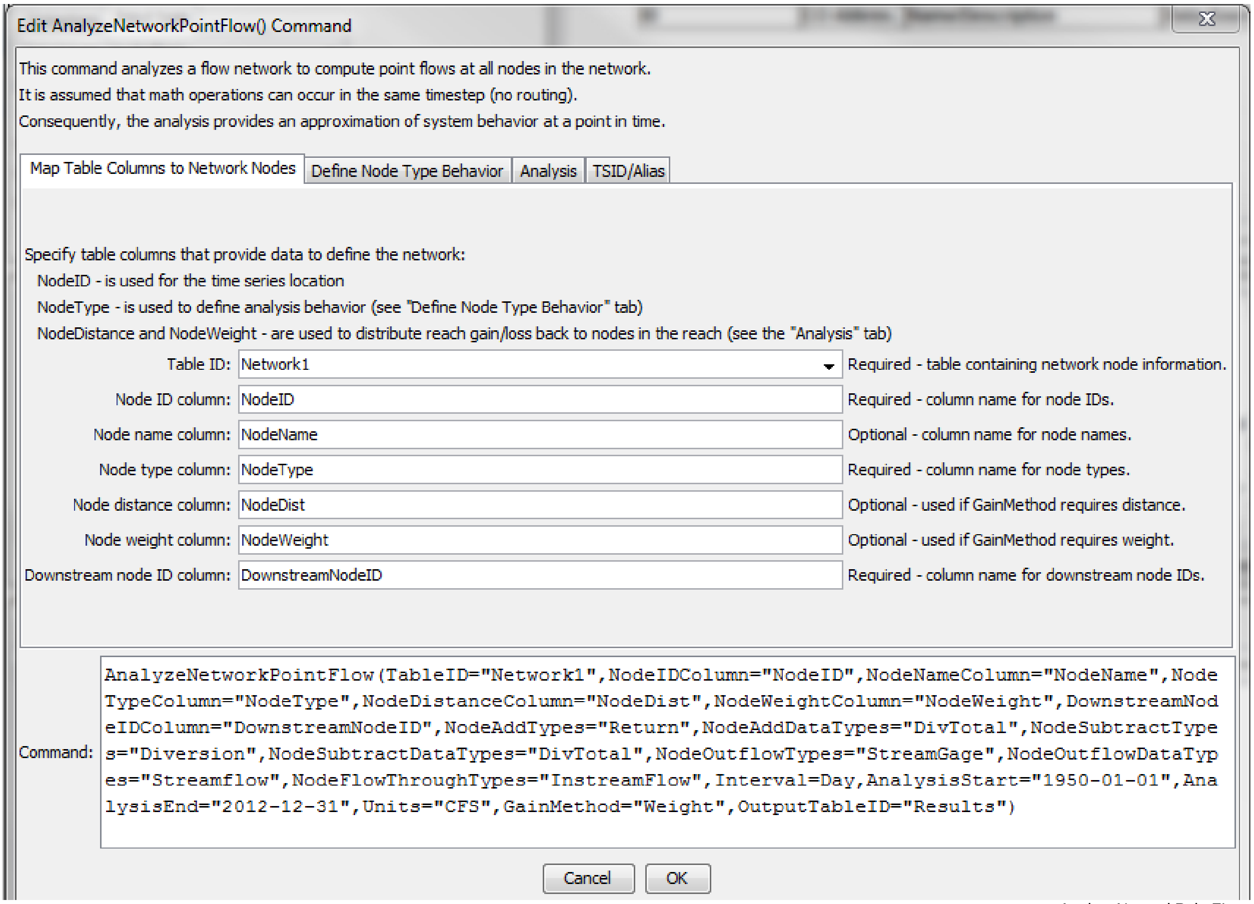
2. Analysis statistics optionally are written to an output table, which contains a row for each network node. Statistics include information such as the number of missing values in the input time series. This information can be used to evaluate the quality of the analysis. **This feature has not yet been implemented.**

Issues that need to be considered include:

1. Missing data in input result in missing data in calculated values. Use TSTool features to fill missing data in time series before using as input to the analysis. Because this may be a major effort, especially for a long analysis period, it may be appropriate to read time series from model data sets. It is envisioned that the output table will provide feedback on how much missing data there is and how it impacts the analysis. To review missing data, use the Period of Record graph or the Data Coverage tool in TSTool.
2. TSTool's graphing tool currently does not allow graphing lines as a step function in the case where no gain/loss is computed. Instead, the line connects the data points. An enhancement to the graphing tool is needed.
3. TSTool does not provide a way to graph a stream reach where the graph values are pulled from each time series for a point in time. Ideally a visualization tool would allow "scrolling" through dates and showing the river reach with flow on the Y axis and node distance on the X axis, although it would be tedious to have to scroll through the period.
4. There may be cases where a subtraction at the node takes all of the flow resulting in a zero or negative value, essentially causing the node to be a known zero point flow. For example, in Colorado, a river call may result in a river drying up during the call. It is possible to estimate when this occurs, but the data quality may be low. Currently TSTool allows negative flows in this case, which indicates that input time series or the simple gain method calculations do not accurately represent the system. One option in this case is to use the TSTool `AdjustExtremes()` command, which maintains mass balance around the extreme values.

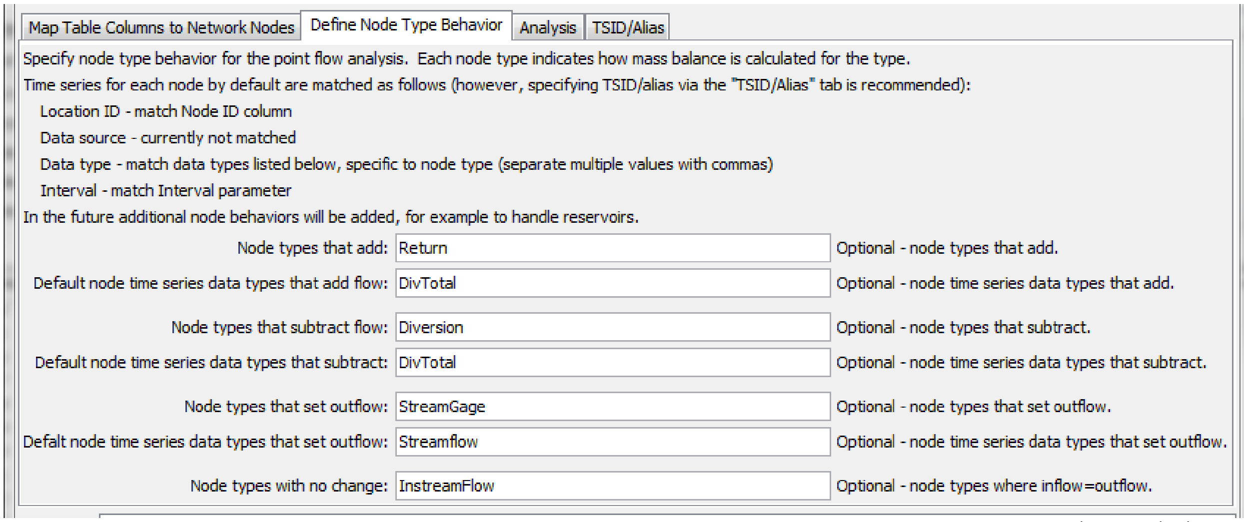
It is important to understand that such a point flow analysis represents a snapshot of the system at any point in time, but does not route flows through the network. Known flows at stream gages are used as fixed values from which other data are estimated. Gains and losses are representative of the network system, essentially interpolating over time and distance. This type of analysis introduces errors in cases where the lag time between nodes would result in significant differences if lagging were considered. In the physical system, changing an upstream flow would result in lagged impacts due to routing; however, the point flow analysis shows the impacts to downstream nodes in the same time step. A more sophisticated model with routing would be needed to represent actual conditions. However, the point flow analysis will be reasonably accurate if gains and losses are occurring because of fairly static phenomena (e.g., groundwater interactions that do not change rapidly within the network travel time). One way to work around these limitations is to use a longer interval, for example monthly instead of daily, in input time series or convert the point flow analysis results.

The following dialog is used to edit the command and illustrates the syntax of the command.



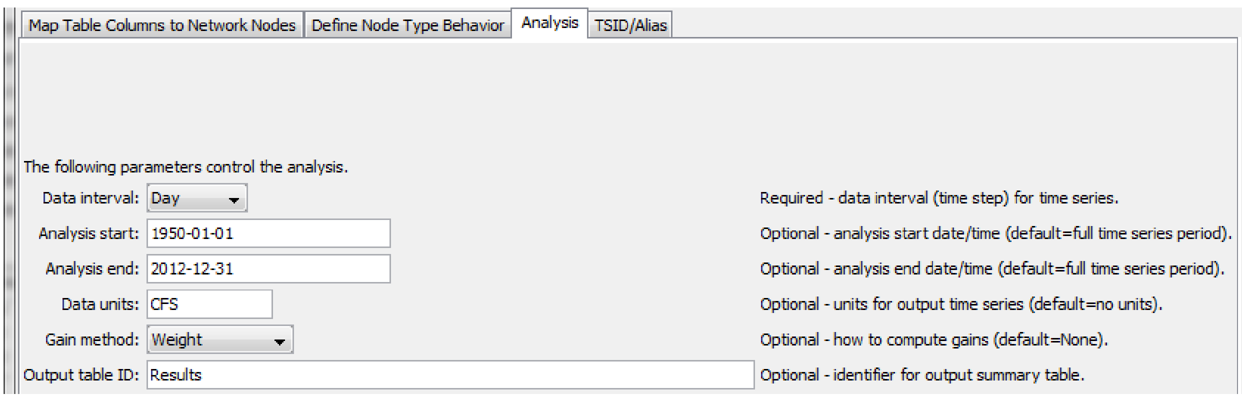
AnalyzeNetworkPointFLow

AnalyzeNetworkPointFlow() Command Editor – "Map Table Columns to Network Nodes" Tab



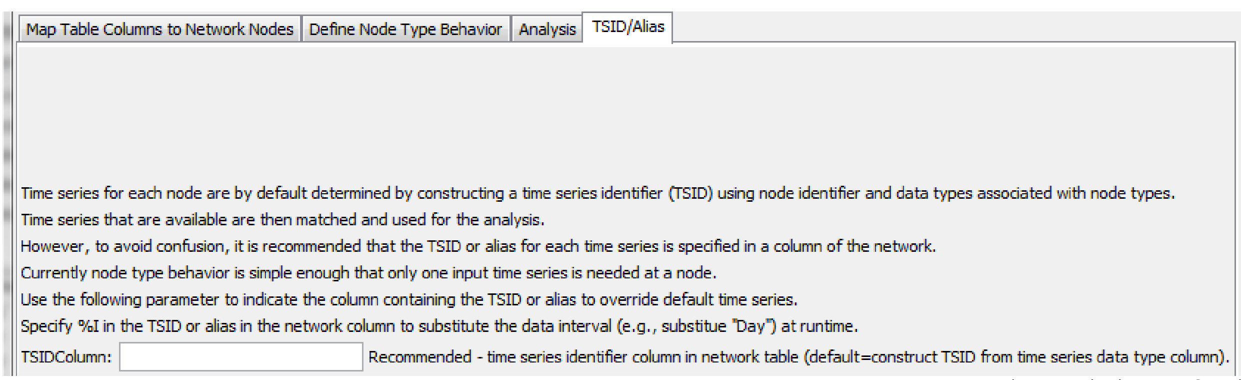
AnalyzeNetworkPointFLow2

AnalyzeNetworkPointFlow() Command Editor – "Define Node Type Behavior" Tab



AnalyzeNetworkPointFLow_AnalysisTab

AnalyzeNetworkPointFlow() Command Editor – "Analysis" Tab



AnalyzeNetworkPointFLow_TSIDTabl

AnalyzeNetworkPointFlow() Command Editor – "TSID/Alias" Tab

The command syntax is as follows:

```
AnalyzeNetworkPointFlow(Parameter=Value,…)
```

Command Parameters

Parameter	Description	Default
TableID	The identifier for the table defining the network.	None – must be specified.
NodeIDColumn	The name of the column in the network table containing node identifiers. Node identifiers will be used for the location ID part of time series identifiers.	None – must be specified.
NodeNameColumn	The name of the column in the network table containing node names.	
NodeTypeColumn	The name of the column in the network table containing node types. The node type is used to specify what calculations will occur for the node.	None – must be specified.
NodeDistance Column	The name of the column in the network table containing node distance. The distance is the measure from the most downstream node and is used when GainMethod=Distance or GainMethod= DistanceWeight.	Must be specified when GainMethod= Distance or GainMethod= DistanceWeight.
NodeWeight Column	The name of the column in the network table containing node weights, which is used to distribute gain/loss when GainMethod=Weight or GainMethod= DistanceWeight (in the latter case the weight is the rate to use).	If not specified when GainMethod=Weight, gain/loss will be distributed evenly for the nodes. Must be specified when GainMethod= DistanceWeight.
Downstream NodeIDColumn	The name of the column in the network table containing downstream node identifiers. This information defines the connectivity of the network.	None – must be specified.
NodeAddTypes	Node types for which time series are added to the node's inflow to compute outflow, for example the Return node type in the above table example. The NodeTypeColumn table column is checked to determine the type for each node in the network.	No additions will occur.
NodeAddDataType	The time series data type to match for the node. The data type is used with the NodeID as the location ID to match available time series to use as input. The TSIDColumn will override the default matching.	No additions will occur.
NodeSubtract Types	Node types for which time series are subtracted from the node's inflow, for example the Diversion node type in the above table example. The NodeTypeColumn table column is checked to determine the type for each node in the network.	No subtractions will occur.
NodeSubtract DataType	The time series data type to match for the node. The data type is used with the NodeID as the location ID to match available time series to use	No subtractions will occur.

Parameter	Description	Default
	as input. The `TSIDColumn` will override the default matching.	
`NodeOutflow Types`	Node types for which time series outflows are set to the node's time input time series, for example the `Streamflow` node type in the above table example. The `NodeTypeColumn` table column is checked to determine the type for each node in the network.	No known flows will be set – gain/loss cannot be computed.
`NodeOutflow DataType`	The time series data type to match for the node. The data type is used with the `NodeID` as the location ID to match available time series to use as input. The `TSIDColumn` will override the default matching.	No subtractions will occur.
`NodeFlow ThroughTypes`	Node types for which time series outflows are set to the node's inflow, for example the `InstreamFlow` node type in the above table example. The `NodeTypeColumn` table column is checked to determine the type for each node in the network.	No known flows will be set – gain/loss cannot be computed.
`Interval`	The time series interval to process. The interval is used with the node identifier and data type to match input time series.	None – must be specified.
`AnalysisStart`	The analysis start, which defines the period for output time series. Specify to a precision consistent with Specify to a precision consistent with Interval.	Global output period.
`AnalysisEnd`	The analysis end, which defines the period for output time series. Specify to a precision consistent with `Interval`.	Global output period.
`Units`	Units for output time series. Warnings will be generated if input time series for the analysis are not consistent with these units.	
`GainMethod`	The method used to prorate the gain/loss between known point flow nodes to other nodes in the reach. Currently this can be used only on non-branching networks. • `Distance` – prorate the gain/loss using distance between nodes (as a portion of the total distance). Use this method if a constant gain/loss rate applies over each reach in the network. • `None` – no gain/loss is estimated, resulting in a discontinuity in an outflow jump above each known point flow. • `DistanceWeight` – prorate the gain/loss using distance*weight as the weight for each node, where the rate is specified in the weight network table column. Use this	`None`

Parameter	Description	Default
	method when the gain/loss rate varies by location and should be represented as a rate. • `Weight` – prorate the gain/loss using the weights specified for each node. Use this method if the gain/loss fraction in a reach is explicitly specified.	
`OutputTableID`	The identifier for the output table to receive analysis results statistics.	No output table will be created.
`TSIDColumn`	The name of the network table column containing time series identifiers or aliases for the input time series for the node. Use `%I` in the column values to replace with the value of the Interval parameter.	Time series will be matched with nodes using the `NodeID` and data types specified with `NodeAddDataType`, etc.

The following command files illustrate how to implement a point flow analysis. In this case the first command file prepares daily time series using the network as input. The time series could similarly be provided by other processing procedures, or read from other model input files. Need to update this example to use alias for the input time series and corresponding TSID column in the network table matching the alias.

```
# Read time series needed to perform the AnalyzeNetworkPointFlow() tests.
# Use data from HydroBase to provide realistic input.
# First read the network table
ReadTableFromExcel(TableID="Network1",InputFile="Network1.xlsx",ExcelColumnNames=FirstRowInRange)
# Get the list of streamflow gages and associated time series
# Free()
CopyTable(TableID="Network1",NewTableID="StreamflowStationList",IncludeColumns="NodeID",
  ColumnMap="NodeID:StreamGageID",ColumnFilters="NodeType:StreamGage")
ReadTimeSeriesList(TableID="StreamflowStationList",LocationColumn="StreamGageID",DataSource="DWR,USGS",
  DataType="Streamflow",Interval="Day",DataStore="HydroBase",IfNotFound=Warn)
WriteDateValue(OutputFile="Network1-StreamGage-Streamflow.dv",MissingValue=NaN,TSList=AllMatchingTSID,
  TSID="*.*.Streamflow.Day.*")
# Get the list of diversion stations and associated time series
Free()
CopyTable(TableID="Network1",NewTableID="DiversionStationList",IncludeColumns="NodeID",
  ColumnMap="NodeID:DiversionID",ColumnFilters="NodeType:Diversion")
ReadTimeSeriesList(TableID="DiversionStationList",LocationColumn="DiversionID",DataSource="DWR",
  DataType="DivTotal",Interval="Day",DataStore="HydroBase",IfNotFound=Warn)
WriteDateValue(OutputFile="Network1-Diversion-DivTotal.dv",MissingValue=NaN,TSList=AllMatchingTSID,
  TSID="*.*.DivTotal.Day.*")
# Get the list of diversion return stations and associated time series
Free()
CopyTable(TableID="Network1",NewTableID="DiversionReturnStationList",IncludeColumns="NodeID",
  ColumnMap="NodeID:DiversionID",ColumnFilters="NodeType:Return")
ReadTimeSeriesList(TableID="DiversionReturnStationList",LocationColumn="DiversionID",DataSource="DWR",
  DataType="DivTotal",Interval="Day",DataStore="HydroBase",IfNotFound=Warn)
WriteDateValue(OutputFile="Network1-Return-
DivTotal.dv",MissingValue=NaN,TSList=AllMatchingTSID,TSID="*.*.DivTotal.Day.*")
```

The second command file performs the point flow analysis. This example is from a TSTool test and fills missing data with a simple approach in order to ensure that no missing values are included in the analysis. A single command file that combines the two command file examples also could be used.

```
# Test analyzing a simple network for point flows
StartLog(LogFile="Results/Test_AnalyzeNetworkPointFlow.TSTool.log")
# Read the network
ReadTableFromExcel(TableID="Network1",InputFile="Data\Network1.xlsx",Worksheet="Network1",
```

```
  ExcelColumnNames=FirstRowInRange)
# Read the time series associated with network nodes (pregenerated)
# Fill diversion time series with zeros so there is something to analyze
# Fill stream gage time series with repeat forward and backward
SetInputPeriod(InputStart="1950-01-01",InputEnd="2013-12-31")
ReadDateValue(InputFile="Data\Network1-Diversion-DivTotal.dv")
ReadDateValue(InputFile="Data\Network1-Return-DivTotal.dv")
FillConstant(TSList=AllMatchingTSID,TSID="*.*.DivTotal.*.*",ConstantValue=0)
ReadDateValue(InputFile="Data\Network1-StreamGage-Streamflow.dv")
FillRepeat(TSList=AllMatchingTSID,TSID="*.*.Streamflow.*.*",FillDirection=Backward)
FillRepeat(TSList=AllMatchingTSID,TSID="*.*.Streamflow.*.*",FillDirection=Forward)
CheckTimeSeries(CheckCriteria="Missing")
# Analyze the network point flow.
AnalyzeNetworkPointFlow(TableID="Network1",NodeIDColumn="NodeID",NodeNameColumn="NodeName",
  NodeTypeColumn="NodeType",NodeDistanceColumn="NodeDist",NodeWeightColumn="NodeWeight",
  DownstreamNodeIDColumn="DownstreamNodeID",NodeAddTypes="Return",NodeAddDataTypes="DivTotal",
  NodeSubtractTypes="Diversion",NodeSubtractDataTypes="DivTotal",NodeOutflowTypes="StreamGage",
  NodeOutflowDataTypes="Streamflow",NodeFlowThroughTypes="InstreamFlow",Interval=Day,
  AnalysisStart="1950-01-01",AnalysisEnd="2012-12-31",Units="CFS",GainMethod="Distance",
  OutputTableID="Results")
```