Dec. 20, 2012

From:
Modelica Association Board
represented by its chairman Prof. Dr.-Ing. Martin Otter

To:
Members of the Modelica Association

Call for Quotation for an Open Source Implementation of the MSL Table Interpolation Blocks

Dear partners,

The Modelica Association plans to finance the development of an open source implementation of the table interpolation blocks utilized in the Modelica Standard Library for a fixed price. Note, currently only the interfaces of these functions are defined and every tool vendor has to provide it's own implementation. Additionally, some missing functionality should be added as outlined below. The required features are listed below. Please, send your quotation (signed, scanned) via email to the MA chairman (Martin.Otter@dlr.de) by January 17, 2013 (he and his organization are excluded from the call). It must contain a firm fixed price, a plan when the developed program will be provided, and the guarantee that the program will be published under the required open source license. Payment will be made within 30 days of delivery of the final product.

The decision for an offer is based on the following criteria (ordered according to priority):

1. Implementation experience for interpolation algorithms.
2. Expected time when the work will be completed
3. Fixed price of the work.
4. Explanation of technical approach.
5. Simplicity to extend the implementation in the future
   (like support for other file formats or other interpolation methods).
6. Validation procedure.
7. Portability of source code to various operating systems

A short quotation of one to two pages is sufficient, summarizing the information relevant for the above criteria. The quotation process will be open: Quotes will be made available to the members of the Modelica Association (via the private Modelica SVN server) after the submission deadline has passed. The selection will be made by the Modelica Association members via electronic voting (persons and organizations that apply for the call are excluded from the electronic voting).

Best regards,

[illegible]
Martin Otter

Technical Requirements of the
Open Source Implementation of the MSL Table Interpolation Blocks

An open source implementation of:

> Modelica.Blocks.Sources.CombiTimeTable
> Modelica.Blocks.Tables.CombiTable1D
> Modelica.Blocks.Tables.CombiTable1Ds
> Modelica.Blocks.Tables.CombiTable2D

shall be provided. Contrary to the current implementation, external objects shall be used (the current implementation uses Integers that are used as index in a statically allocated array that contains the pointers to the table data). The implementation must be performed in Modelica and C. Modelica code must be provided under Modelica License 2. C-code must be provided under the new BSD license (http://opensource.org/licenses/BSD-3-Clause).

The current interface to the tables is:

Modelica.Blocks.Sources.CombiTimeTable

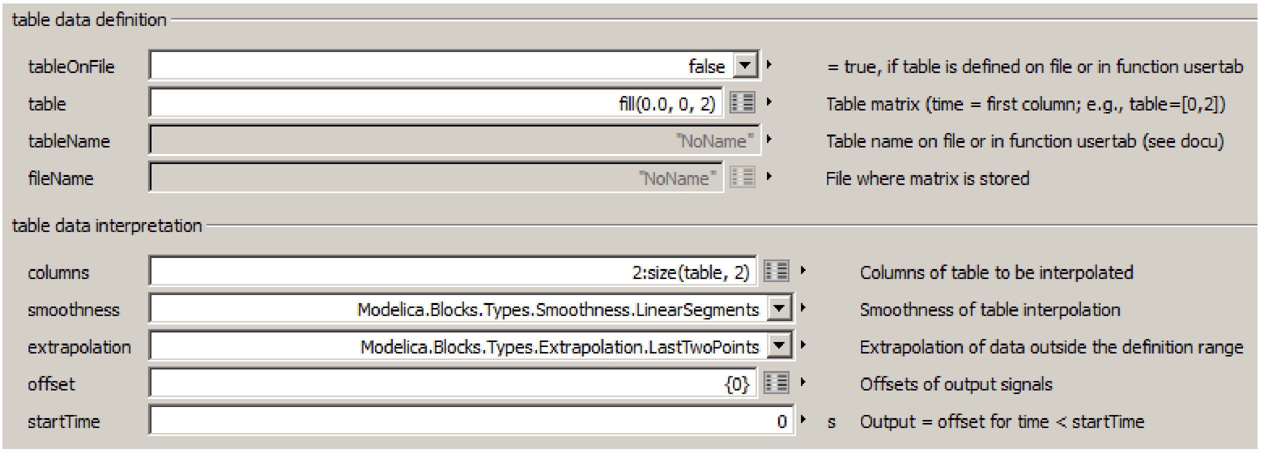

table data definition

tableOnFile	false		= true, if table is defined on file or in function usertab
table	fill(0.0, 0, 2)		Table matrix (time = first column; e.g., table=[0,2])
tableName	"NoName"		Table name on file or in function usertab (see docu)
fileName	"NoName"		File where matrix is stored

table data interpretation

columns	2:size(table, 2)		Columns of table to be interpolated
smoothness	Modelica.Blocks.Types.Smoothness.LinearSegments		Smoothness of table interpolation
extrapolation	Modelica.Blocks.Types.Extrapolation.LastTwoPoints		Extrapolation of data outside the definition range
offset	{0}		Offsets of output signals
startTime	0	s	Output = offset for time < startTime

Modelica.Blocks.Tables.CombiTable1D

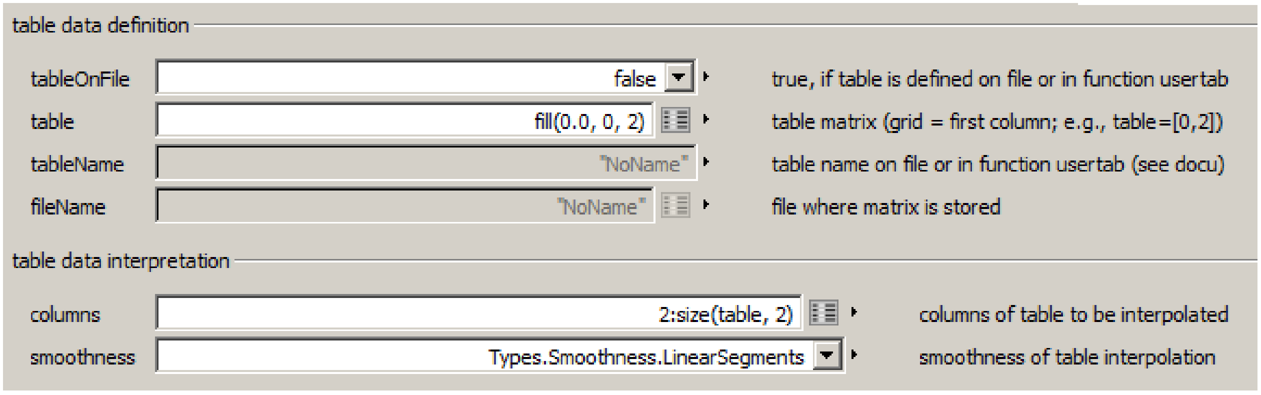

table data definition

tableOnFile	false	true, if table is defined on file or in function usertab
table	fill(0.0, 0, 2)	table matrix (grid = first column; e.g., table=[0,2])
tableName	"NoName"	table name on file or in function usertab (see docu)
fileName	"NoName"	file where matrix is stored

table data interpretation

columns	2:size(table, 2)	columns of table to be interpolated
smoothness	Types.Smoothness.LinearSegments	smoothness of table interpolation

Modelica.Blocks.Tables.CombiTable1Ds

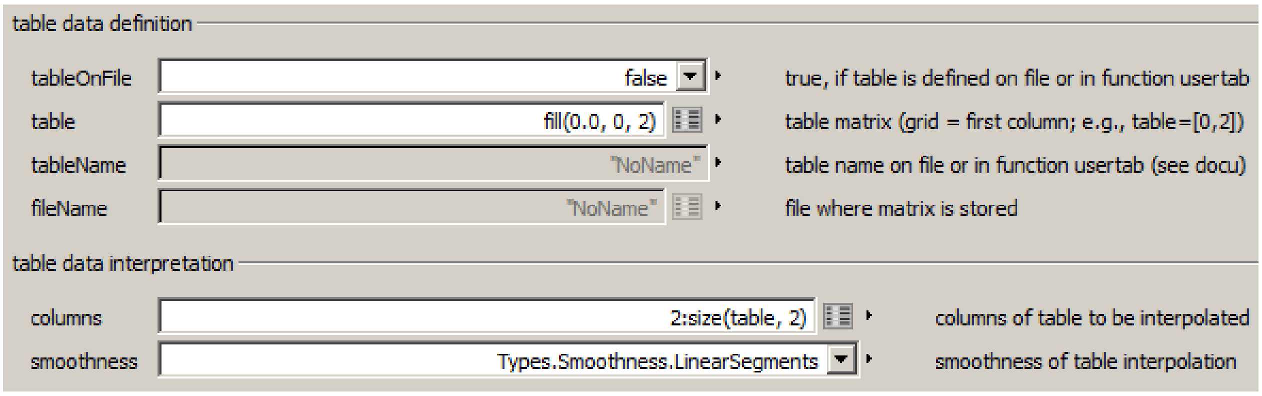

table data definition

tableOnFile	false	true, if table is defined on file or in function usertab
table	fill(0.0, 0, 2)	table matrix (grid = first column; e.g., table=[0,2])
tableName	"NoName"	table name on file or in function usertab (see docu)
fileName	"NoName"	file where matrix is stored

table data interpretation

columns	2:size(table, 2)	columns of table to be interpolated
smoothness	Types.Smoothness.LinearSegments	smoothness of table interpolation

Modelica.Blocks.Tables.CombiTable2D

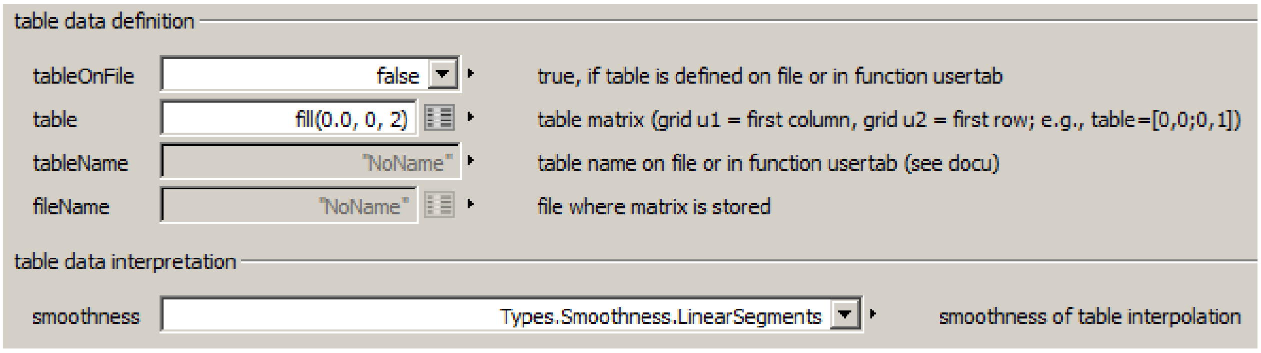

table data definition

tableOnFile	false	true, if table is defined on file or in function usertab
table	fill(0.0, 0, 2)	table matrix (grid u1 = first column, grid u2 = first row; e.g., table=[0,0;0,1])
tableName	"NoName"	table name on file or in function usertab (see docu)
fileName	"NoName"	file where matrix is stored

table data interpretation

smoothness	Types.Smoothness.LinearSegments	smoothness of table interpolation

The implementation must take into account the following requirements:

- If option tableOnFile = true, then the data is read from file, must be stored in C-storage and must not be reported to Modelica (in order that large data arrays can be handled without the overhead of Modelica variables).

- C-storage for data read from file is dynamically allocated. Data that is directly stored in function usertab (siehe documentation of the table blocks) is statically allocated and defined in this function.

- All abscissa values must be monotonically increasing for CombiTimeTable and strict monotonically increasing for the other tables. During initialization, this assumption must be checked.

- Searching the interpolation interval for a given abscissa value must be performed in an efficient way (for example by storing the last search interval and performing a linear search, or by performing always a binary search).

- At least the data formats on file must be supported that are supported in Dymola (ASCII-format, or Matlab 4 binary format).

- The first derivative of the outputs with respect to the inputs must be provided as Modelica functions for all table types. These functions must be defined with the derivative annotation in the respective interpolation function. The first derivative functions are only supported for smooth interpolation methods (that is, the first derivative must be continuous).
- Special features in combiTimeTable:
  The data points can be discontinuous (identified by identical time instants). In that case a time event must be triggered at the discontinuous points, but not at the other data points. A portable implementation can perform this in a similar way as the pure Modelica implementation of Modelica.Blocks.Sources.TimeTable (by having the time event handling in

Modelica and by performing the interpolation of the actual segment of strict monotonically increasing values in C).

- All tables have parameter „smoothness“ with the following meaning:

```
type Smoothness = enumeration(
  ConstantSegments "Table points are not interpolated, but the value from the previous
                    abscissa point is returned",
  LinearSegments "Table points are linearly interpolated",
  ContinuousDerivative  "Table points are interpolated such that the
                 first derivative is continuous)
```

  "ContinuousDerivative" shall be implemented with Akima Splines:
    - For 1D-interpolation with univariate Akima Splines (see for example the open source implementation in Matlab under BSD license: http://www.mathworks.se/matlabcentral/fileexchange/1814-akima-interpolation)
    - For 2D-interpolation with bivariate Akima Splines (see for example the open source implementation in Fortran 90 under GPL license: http://sosie.sourceforge.net/).

  For the CombiTimeTable block the following (read only) parameters have to be implemented:

```
  final parameter Real t_min(fixed=false)
    "Minimum abscissa value defined in table";
  final parameter Real t_max(fixed=false)
    "Maximum abscissa value defined in table";
```

  Finally, for the CombiTimeTable block, an extrapolation option "extrapolation" shall be added with the possible values:

```
type Extrapolation = enumeration(
    NoExtrapolation "Extrapolation triggers an error"
    HoldLastPoint   "Hold the last table point outside of the
                     table scope",
    LastTwoPoints   "Extrapolate linearly through the last two table
                     points outside of the table scope"
    Periodic        "Repeat the table scope periodically.
                     Independent variable is modified as follows:
                     mod(x-xmin,xmax-xmin)+xmin")
```

  Default is "LastTwoPoints" (since this is the default in MSL 3.2)

Implementation Suggestions

The following notes are suggestions for implementation, but it is not required that the implementation is performed in this way (however, it is one way to fulfill criteria 5: "Simplicity to extend the implementation in the future"):

1. The core interpolation functions should be stored in the ModelicaServices package, and a reference implementation should be provided. This gives tool vendors the freedom to replace the reference implementation by an own implementation.

2. The functionality is implemented completely in Modelica, with exception of reading data from file. For the latter, the free packages "ScanTextFile" and "ScanBinaryFile" are interfaces to a corresponding C-implementation. They are available in package Modelica_NewTables on the svn server: https://svn.modelica.org/projects/Modelica/branches/development/NewTables. These packages have the advantage that reading data from textual or binary file format can be implemented in a simple and quick way, and the implementation provides good error

diagnostics in case of failure (e.g. printout of the line where the error occurred together with file name, line number and character number).
The efficient handling of large arrays is performed by introducing an annotation "FunctionArray = true". The annotation informs the Modelica tool, that the array is planned to be used only as input and/or output argument for Modelica functions and that the array should not be stored in a result file. Therefore, a tool need not split the array in its elements but can handle it as one piece of memory that is passed between function calls. Furthermore, the dimensions of the array can be fixed during initialization (and need not to be known during translation). If this annotation is not supported by a tool, there might be restrictions (e.g. the table date on file must be known during translation) and the handling of large arrays might become inefficient.