# A new Implementation of the N-D Lookup Tables

Torsten Sommer Markus Andres Stephan Diehl
Modelon GmbH
Agnes-Pockels-Bogen 1
D-80992 Munich, Germany
torsten.sommer@modelon.com markus.andres@modelon.com stephan.diehl@modelon.com

## Abstract

The HDF5Table library is an open-source solution for the efficient handling, exchange and interpolating access of typical data sets in system simulation. The library consists of C-functions, python scripts and examples and can be used with different applications like Modelica or Simulink. Furthermore a comprehensive set of tools that allows the user to create, migrate, edit, compare and manage the datasets is available.

The application range covers data import from measurements or other simulations, integration of datasets in preprocessing routines, the usage of the datasets in the simulation and the post processing of simulation results. To eliminate a major source of errors after data exchange between simulation tools or different companies and to validate the datasets each dataset can have a physical unit and quantity attached to it. The table data can be easily accessed with different methods for inter- and extrapolation. To persist and exchange the data sets a subset of the HDF5 standard is used. With the HDF5 API the data access is fast for large files with many variables containing millions of values and the datasets can be opened in many other tools.

*Keywords: HDF5; lookup tables; unit and quantity safety; interpolation; extrapolation*

## 1 Introduction

Lookup tables traditionally play a major role in industrial simulations. They are used in a wide range of applications where physical models or parameters are not available or the evaluation of the existing models is computationally too expensive. Real-time simulations and hardware-in-the-loop setups are two prominent examples. In these systems lookup tables are used to re-play recorded stimulus from measurements as well as pre- and post-processed data from test benches. Another application is pre-calculated lookup tables from long running system or finite element simulations.

A number of solutions exist for Modelica and other simulation platforms some of which are discussed in detail in the following section. All of these solutions suffer from different limitations and problems the proposed implementation together with a set of supporting tools is trying to solve.

## 2 Existing Solutions

The Modelica Standard Library (MSL) provides a number of tables in its Blocks Package. A general separation is done based on the provided input functionality. Here the prefix Combi shows the capability of the table to either take direct user input in the form of parameters from the Modelica environment or to read data from files. In industrial applications the second option is nearly solely used as it is more convenient even for tables of modest size.

Table blocks for different needs are provided and therefore are split into the Blocks.Sources and the Blocks.Tables package. The separation into the packages is done as the tables in the Sources package implicitly define the single input of the table as the simulation time. These tables are mainly used for the playback of recorded continuous stimulus e.g. steering input or velocity signals over time. As these tables have a single input and are well suited for time series simulation they are not in the focus of this paper.

This leads to the Blocks.Tables package. These tables are often used to cover effects that cannot be modeled based on physics with reasonable effort or performance. Therefore tables are used to approximate the behavior depending on a varying number of inputs. Still for the tables in Blocks.Tables the number of inputs is limited to one or two which becomes a limiting factor in many applications. Therefore Dymola provides the DataFiles package that contains the TableND which extends the number of inputs to a theoretically unlimited amount. As indicated by the missing prefix, data can only be read from .mat files. The DataFiles Package contains a number of functions that support the user with the generation of the necessary .mat files. Additionally some Matlab scripts are provided to support the data storing process.

Now at first glance for Dymola users there is the possibility to have capable tables in their simulations using either the MSL's Tables package or the DataFiles package. Still there are a couple of limitations to that which we want to point out now. The TableND's data format is limited to version 4 .mat files which limits the names in the file to 19 characters and – much more of interest here – limits the number of possible dimensions stored in the files to two. Therefore the multi-dimensional tables cannot be stored in their natural format but have to be converted. This problem is overcome by generating three vectors by convention named dim, grid and table. The length of the vector dim equals the number of dimensions of the table each entry indicating the amount of scale values in that dimension. In the variable "grid" the scales for all dimensions are stored implying that the length of grid has to be equal to the sum of the elements in dim. Finally in the table variable finally all values of the multidimensional table are stored in one vector. All of this makes it very hard to interpret the values in the table without suitable conversion scripts.

Another disadvantage of this solution is the different behavior of CombiTables and TableND when it comes to extrapolation. Whereas the CombiTables extrapolate linearly the TableND keeps values constant when exiting the defined area. For version 3.2.1 of the MSL efforts have been completed to make the implementation of the MSL tables open source and to enhance their functionality. New features include that outputs of tables can now be differentiated once, and newer .mat File formats are supported [7]. Still that does not have any influence on the limiting factors mentioned before, as this is only valid for the tables contained in the MSL.

## 3 Goals of the new Implementation

The goal of the proposed implementation is to come up with a library that provides a superset of the features of the existing solutions that is entirely open source and accompanied by comprehensive set of tools to leverage its functionality and ease the transition from existing solutions. The scope of possible applications for both tables and tooling goes beyond Modelica models since the tables can be easily ported to other simulation platforms (e.g. Simulink, CarMaker) or integrated into custom C-based solutions. This takes the concept of separation of model and data to a higher level as data can be used independently from the simulation tool.
Furthermore one of the basic concepts of Modelica, namely unit and quantity safety is adapted for table-based data. Every dataset regardless of its dimensions can have a physical unit and quantity (and other attributes) attached to it which allows the table block to validate the data upon loading and eliminates a major source of errors that arises from incompatible data being used in the simulations.
Future versions of the library will also be able to record signals from a simulation and provide support for 'compiled-in' data which is especially useful for platforms that do not have a file system or to protect the intellectual property contained in the data.

## 4 Features

To provide a one-stop solution for all uses of table based data the table block features different methods to inter- and extrapolate the sampled data which are briefly presented in the following sections.

### 4.1 Interpolation

Generally two major uses of tables in models can be distinguished: the first is the playback of recorded continuous stimulus e.g. steering input or discrete data like bus events. Both types are usually one-dimensional and have the time as abscissa.

The second major use case for tables is found in models that cannot be physically modeled and thus use measured data to approximate their behavior or are too complex to be simulated e.g. in a real-time context. These table-based models usually depend on more than one parameter and thus require multi-dimensional tables. A prominent example in which both types of tables can be found are hardware-in-the-loop test benches.

The HDF5Table block provides three interpolation methods that can be configured on a per-instance basis: 'Hold', 'Linear' and 'Akima'.

'Hold' will simply return the previous value in the respective dimension and can be used to playback time discrete data where interpolation does not make sense e.g. the selected gear in a vehicle.

'Linear' interpolates linearly between the neighboring sample points in each dimension.

'Akima' uses a spline based interpolation method proposed by Akima [5] where the interpolated function passes through the sample points and its first derivative is continuous.

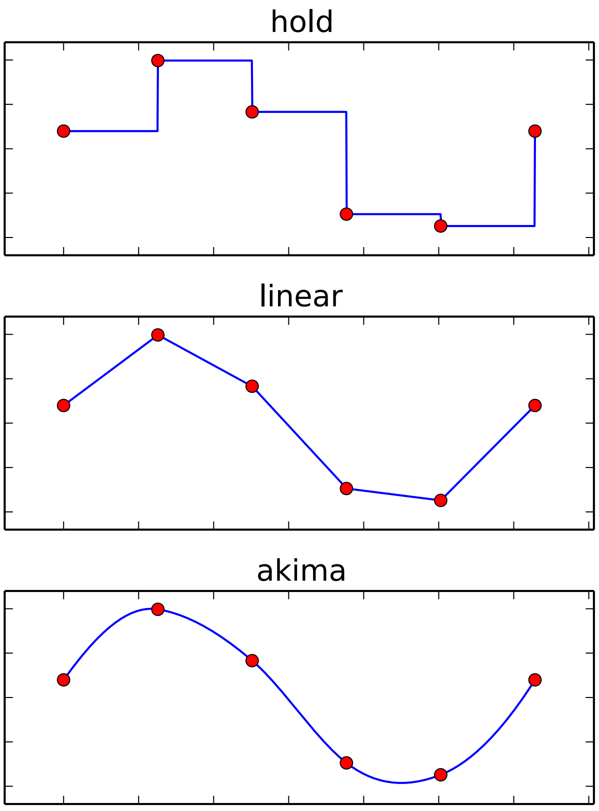


**Figure 1 Interpolation Methods**

This has a positive impact on the stability of models that use the derivative in their calculations. Figure 1 shows the three interpolation methods applied to a one-dimensional dataset with six equally spaced sample points of a sine function.

## 4.2 Extrapolation

Five methods are provided for extrapolation:

- Hold
- Linear
- Loop
- PingPong
- None

Similar to the interpolation 'Hold' simply repeats the last sample value in the respective dimension. 'Linear' uses the last two samples to linearly extrapolate. For the playback of time series it is often desired to infinitely repeat a given set of samples. 'Loop' repeats the recorded signal using the selected interpolation method.
The 'PingPong' method works similarly to the loop method but instead of starting over it goes back and forth along the respective axis. This method is especially useful for lookup tables used for devices that have a symmetric characteristic field like electric machines. To better understand this method consider the following example of the loss characteristics of a permanent magnet synchronous machine (PMSM) given as the power losses versus rotational speed as the abscissa and the torque as the ordinate. Three different types of lookup tables are common for PMSMs: first quadrant, first and second quadrant and all four quadrants corresponding to the motor, motor and generator and motor and generator for positive and negative rotation directions. With the 'PingPong' method one table block can be used for all three types of tables without conditional instantiation or additional logic inside the model. Figure 2 shows the extrapolated curve of the five methods for a one-dimensional dataset with four equally spaced and linearly increasing samples.
The extrapolation method "None" disables extrapolation and raises an error when requested value is outside the range of the table.

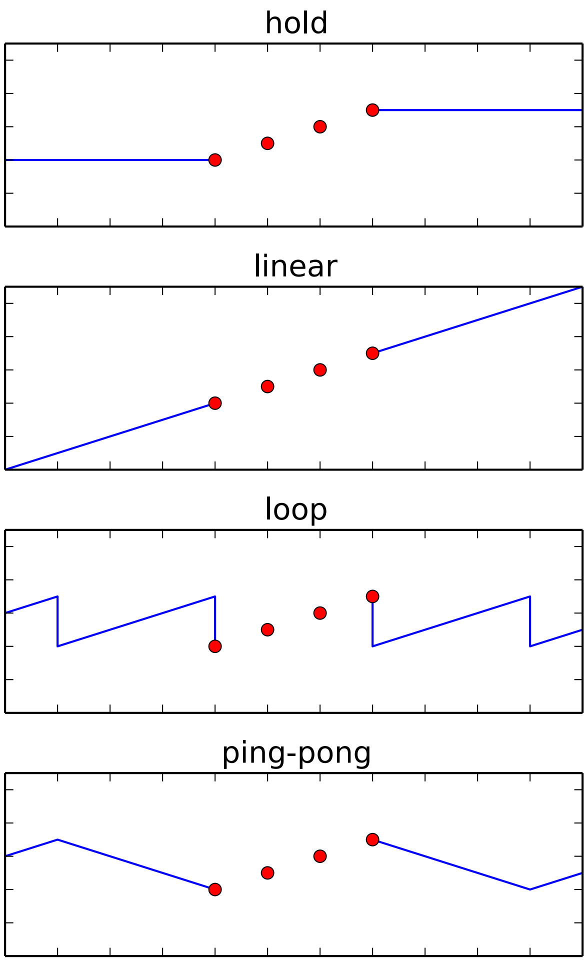


**Figure 2 Extrapolation Methods**

## 5 HDF5 File Structure

In order to efficiently store and exchange the data for the tables a subset of the HDF5 standard is used. Every table is stored as n-dimensional dataset of type Float64. The scales for each dimension are stored as separate datasets. For every dimension one scale dataset is attached to the table dataset using the HDF5 dimension scales API [3].

To store metadata like the physical units and quantities a set of attributes has been defined that is evaluated when reading the datasets. The following table lists the attribute names together with the required data type and an example value.

| Attribute Name | Type | Example |
|---|---|---|
| QUANTITY | String | "AngularVelocity" |
| UNIT | String | "rad/s" |
| DISPLAY_UNIT | String | "1/min" |

QUANTITY and UNIT denote the physical quantity and unit of the stored data. DISPLAY_UNIT is the unit that is used to display the values to the user e.g. when editing them. The literals used to represent the units and quantities are the same as in Modelica and can be found in the Modelica.SIunits and Modelica.SIunits.Conversions.NonSIunits packages of the MSL.

Scales must always be one-dimensional and strictly monotonic increasing and there must be at most one scale attached per dimension whose length matches the extent of the dataset in the respective dimension. For a three-dimensional dataset with extent 2×3×4 the scale for the second dimension must have exactly three values.

Users may add custom attributes and datasets to store additional metadata, documentation or results and use groups [6] to structure the datasets.

## 6 Modelica

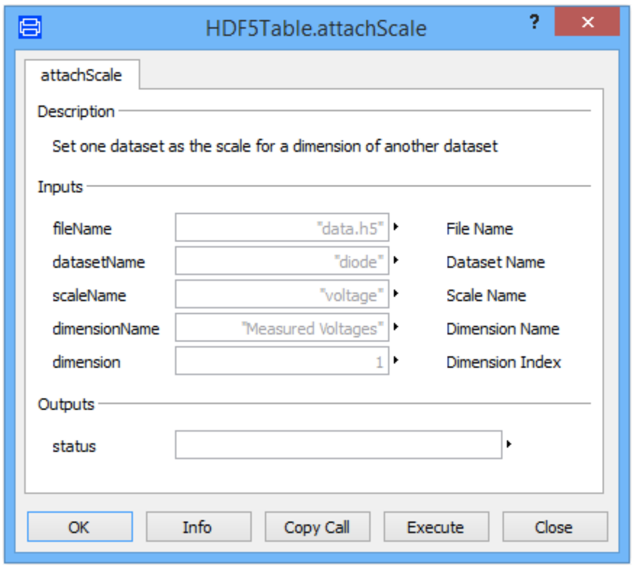
The HDF5Table Modelica library comes as a .mo file that contains the blocks, functions and examples together with the C header files and pre-compiled object libraries for the table and HDF5 as a ready-to-use package. The Modelica functions and blocks are presented in detail in the following sections.

### 6.1 Functions

To directly read and write scalars, vectors and matrices from and to HDF5 files the library includes the following functions that are similar to the readMAT* / writeMAT* functions in the DataFiles package that ships with Dymola:

- writeVector
- writeMatrix
- attachScale
- readScalar
- readScalarChecked
- readVector
- readVectorChecked
- readMatrix
- readMatrixChecked
- attachScale

The "checked" versions of the functions take the expected unit and quantity as an additional parameter and return an error code if the parameters do not match the unit or quantity stored in the loaded dataset.
The attachScale function attaches a dataset as the scale for the dimension of another dataset. All functions are realized as external C code that can be linked into the model or be called directly using the simulation tool as shown in the figure below.


HDF5Table.attachScale

attachScale

Description

Set one dataset as the scale for a dimension of another dataset

Inputs

| | | |
|---|---|---|
| fileName | "data.h5" | File Name |
| datasetName | "diode" | Dataset Name |
| scaleName | "voltage" | Scale Name |
| dimensionName | "Measured Voltages" | Dimension Name |
| dimension | 1 | Dimension Index |

Outputs

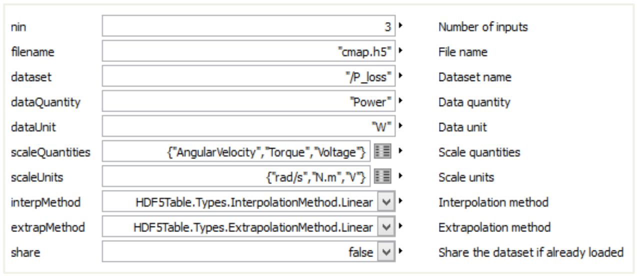
status

**Figure 3 attachScale Function Dialog**

The library can also read / write higher dimensional arrays (up to 32 dimensions). Entering higher dimensional arrays however can be cumbersome, as e.g. Dymola does not provide UI support for editing values with more than two dimensions.

### 6.2 Blocks

The core of the library is the NDTable block that loads and interpolates the data during the simulation. It takes the file name of the HDF5 file, the dataset name and optionally the expected units and quantities of the dataset and scales as input and returns the inter- or extrapolated value for every time step. Figure 4 shows the parameters of the NDTable block with all quantities and units set for a dataset that holds the power losses of an electric machine as a function of rotational speed, torque and source voltage.



| | | |
|---|---|---|
| nin | 3 | Number of inputs |
| filename | "cmap.h5" | File name |
| dataset | "/P_loss" | Dataset name |
| dataQuantity | "Power" | Data quantity |
| dataUnit | "W" | Data unit |
| scaleQuantities | {"AngularVelocity","Torque","Voltage"} | Scale quantities |
| scaleUnits | {"rad/s","N.m","V"} | Scale units |
| interpMethod | HDF5Table.Types.InterpolationMethod.Linear | Interpolation method |
| extrapMethod | HDF5Table.Types.ExtrapolationMethod.Linear | Extrapolation method |
| share | false | Share the dataset if already loaded |

**Figure 4 Parameters of the NDTable block**

The DatasetRecorder block takes the scales as parameters and sweeps successively over the volume spanned by the scales by applying the actual scale values to the vector output. When a rising edge is detected on the trigger port it records the value on the in-port and applies the next set of scale values to the input until all samples in the volume have been recorded. Using this block tables with a large number of sample points can be generated without the overhead of restarting the simulation which can save a considerable amount of time.

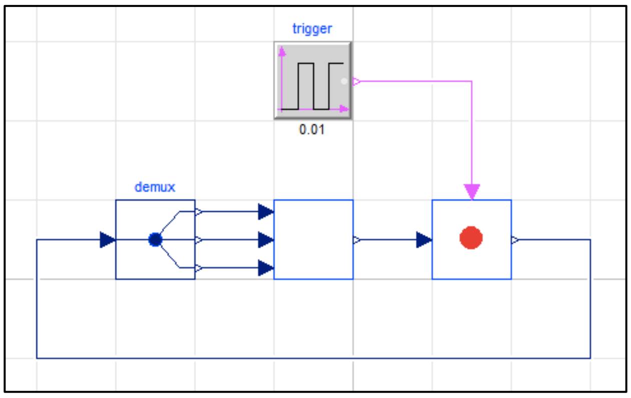


**Figure 5 RecorderBlock Example Model**

Figure 5 shows the RecordTable example model where the DatasetRecorder block is connected to a dummy model that has three in-ports and whose output is recorded upon every rising flank of the trigger.

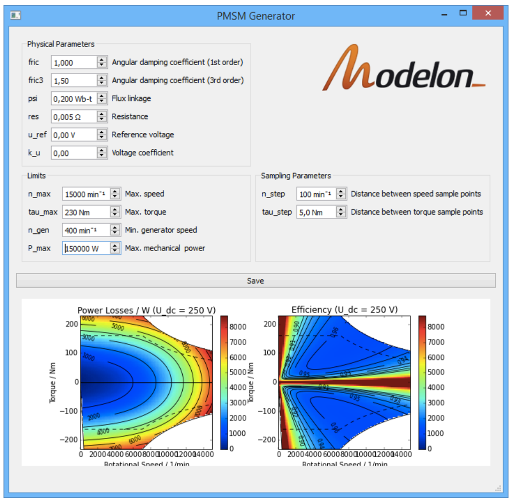
## 7 Matlab / Simulink

Matlab and Simulink are two of the most commonly used simulation and scripting environments. Therefore the HDF5Table library also includes a Simulink S-Function and Matlab scripts that allow users to reuse their existing datasets without changes on this platform. The Simulink table block has the same inter- and extrapolation methods as the Modelica library and the underlying S-Functions are based on the same C-sources which can be a major advantage when porting models to this platform that rely on the specific behavior of the implementation. Just like the Modelica library the Simulink library ships as a precompiled shared library that can be instantly used in Simulink.

## 8 Tooling

To leverage the tables in the simulation they are accompanied by a comprehensive set of tools that allow the user to create, migrate, edit, compare and manage the datasets. All tools are included in an Eclipse distribution that is used as an integration platform but can also be obtained and used separately.

The tools include a Python library based on matplotlib [8] and NumPy [9] to read, write, manipulate and plot data that can be debugged and run directly from within Eclipse using the PyDev environment [10]. Python's support for a large number of data formats allows the user to write import and export scripts for their existing data with minimal effort based on the provided examples. With the included Python scripts both the text based and binary data files used by the Blocks.Tables and Datafiles blocks can be converted to HDF5.



Physical Parameters

| | | |
|---|---|---|
| fric | 1,000 | Angular damping coefficient (1st order) |
| fric3 | 1,50 | Angular damping coefficient (3rd order) |
| psi | 0,200 Wb-t | Flux linkage |
| res | 0,005 Ω | Resistance |
| u_ref | 0,00 V | Reference voltage |
| k_u | 0,00 | Voltage coefficient |

Limits

| | | |
|---|---|---|
| n_max | 15000 min⁻¹ | Max. speed |
| tau_max | 230 Nm | Max. torque |
| n_gen | 400 min⁻¹ | Min. generator speed |
| P_max | 150000 W | Max. mechanical power |

Sampling Parameters

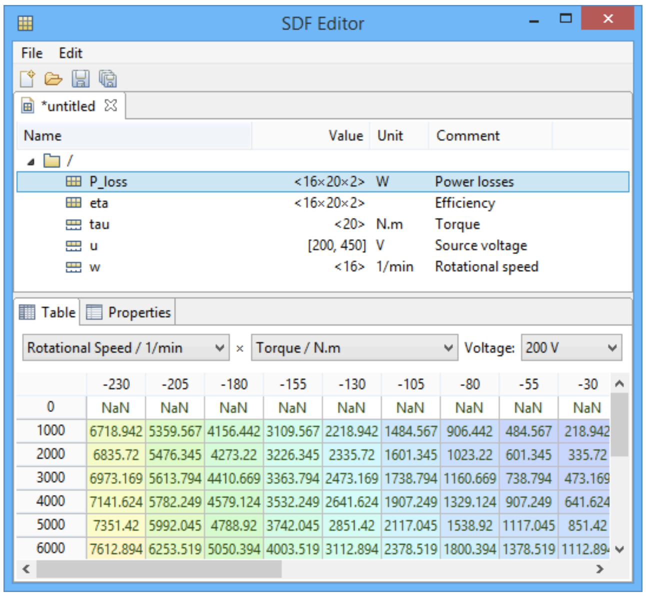
| | | |
|---|---|---|
| n_step | 100 min⁻¹ | Distance between speed sample points |
| tau_step | 5,0 Nm | Distance between torque sample points |

**Figure 6 Python script with graphical UI**

Figure 6 shows an example Python script with a graphical user interface that allows the user to create characteristic maps of electric machines for use with the NDTable block and that can serve as a basis for custom HDF5 generators or processors.

Furthermore an EMF [11] based editor is included that allows the user to conveniently view, edit and validate the data files. Finally the datasets can be compared and merged with the included EMF compare editor [12].

The figure below shows the HDF5 editor viewing the content of a three-dimensional characteristic map dataset with scales.



| Name | Value | Unit | Comment |
|---|---|---|---|
| / | | | |
| P_loss | <16×20×2> | W | Power losses |
| eta | <16×20×2> | | Efficiency |
| tau | <20> | N.m | Torque |
| u | [200, 450] | V | Source voltage |
| w | <16> | 1/min | Rotational speed |

Rotational Speed / 1/min × Torque / N.m Voltage: 200 V

| | -230 | -205 | -180 | -155 | -130 | -105 | -80 | -55 | -30 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 1000 | 6718.942 | 5359.567 | 4156.442 | 3109.567 | 2218.942 | 1484.567 | 906.442 | 484.567 | 218.942 |
| 2000 | 6835.72 | 5476.345 | 4273.22 | 3226.345 | 2335.72 | 1601.345 | 1023.22 | 601.345 | 335.72 |
| 3000 | 6973.169 | 5613.794 | 4410.669 | 3363.794 | 2473.169 | 1738.794 | 1160.669 | 738.794 | 473.169 |
| 4000 | 7141.624 | 5782.249 | 4579.124 | 3532.249 | 2641.624 | 1907.249 | 1329.124 | 907.249 | 641.624 |
| 5000 | 7351.42 | 5992.045 | 4788.92 | 3742.045 | 2851.42 | 2117.045 | 1538.92 | 1117.045 | 851.42 |
| 6000 | 7612.894 | 6253.519 | 5050.394 | 4003.519 | 3112.894 | 2378.519 | 1800.394 | 1378.519 | 1112.89 |

**Figure 7 HDF5 Editor**

In the editor values are displayed in "display units" and can be conveniently entered as expressions like "sin(pi/4)" to get the sine of a 45 degree angle or "ones(2, 3)" to get a 2×3 matrix of ones. All operations performed in the editor like copy & paste, move, delete etc. can be undone with the standard shortcuts or menu items. Drag & drop is supported to move datasets between groups and even between files.

## 9 Future Work

Currently the library is developed in Visual Studio 2010 for Windows. It is planned to port the library to the hardware and software platforms listed below.

Hardware Platforms:

- Windows 32 and 64 bit
- Linux 32 and 64 bit
- dSPACE SCALEXIO
- dSPACE DS1006

Compilers:

- Microsoft compilers (VC6 and ≥ VS2005 (Win32 and x64))
- MinGW (GCC 4.4.0 and GCC 4.7.2)
- Cygwin (GCC 4.3.0)
- GCC 4.x on Linux

We are also planning to further extend the tooling and to include a comprehensive documentation and examples that showcase common uses and best practices. A future version of the library will include additional inter- and extrapolation methods (e.g. boundary slope extrapolation for the Akima method) and support for the Der() derivative of the interpolated values to reduce simulation time and improve accuracy [1].

## 10 Conclusions

The implemented library and its extensions show that most features of the existing tables in the MSL's Blocks.Tables and the DataFiles package can be combined in a single table simplifying the application for the user. Additionally the error-proneness of the overall simulation process can be reduced substantially based on unit and quantity checks.
The use of open standards like HDF5 and Modelica guarantee that this open-source implementation is expandable and can be customized for different needs including wider tool support. To enable fast adaption of the presented library a set of tools is provided enabling the user to quickly generate new or migrate existing datasets.

## References

[1] Modelica Language Specification Version 3.3, Section 12.7, https://www.modelica.org/documents/ModelicaSpec33.pdf

[2] Call for Quotation of an Open Source Implementation of the MSL Table Interpolation Blocks, https://www.modelica.org/news_items/call-texts-to-improve-modelica-2012/2012-12-20-Call-for-quotation-for-MSL-tables.pdf/at_download/file

[3] HDF5 Dimension Scale API Reference, http://www.hdfgroup.org/HDF5/doc/HL/RM_H5DS.html

[4] Proposal for a Standard Time Series File Format in HDF5, http://www.bausch-gall.de/ecp12076495_PfeifferBausch-GallOtter.pdf

[5] "A new method of interpolation and smooth curve fitting based on local procedures", Journal of ACM 17, 4 (1970), 589-602

[6] HDF5 Group Interface, http://www.hdfgroup.org/HDF5/doc/RM/RM_H5G.html

[7] Modelica Newsletter by Martin Otter https://www.modelica.org/publications/newsletters/2013-2#item227

[8] matplotlib, http://matplotlib.org/

[9] NumPy, http://www.numpy.org/

[10] PyDev, http://pydev.org/

[11] Eclipse Modeling Framework, http://www.eclipse.org/modeling/emf/

[12] EMF Compare, http://www.eclipse.org/emf/compare/