Introduction to mdtsObject

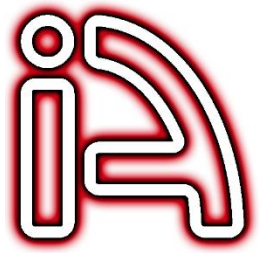
First steps to use mdts Objects

Authors: Roland Ritt

Date: 06.08.2018

Document type: Internal Presentation

Document no.:



Chair of Automation
University of Leoben, Peter-Tunner-Strasse 27, A-8700 Austria
Tel: 0043 3842 402 5301, FAX: 0043 3842 402 5302
automation@unileoben.ac.at, http://automation.unileoben.ac.at

MONTAN UNIVERSITÄT
WWW.UNILEOBEN.AC.AT

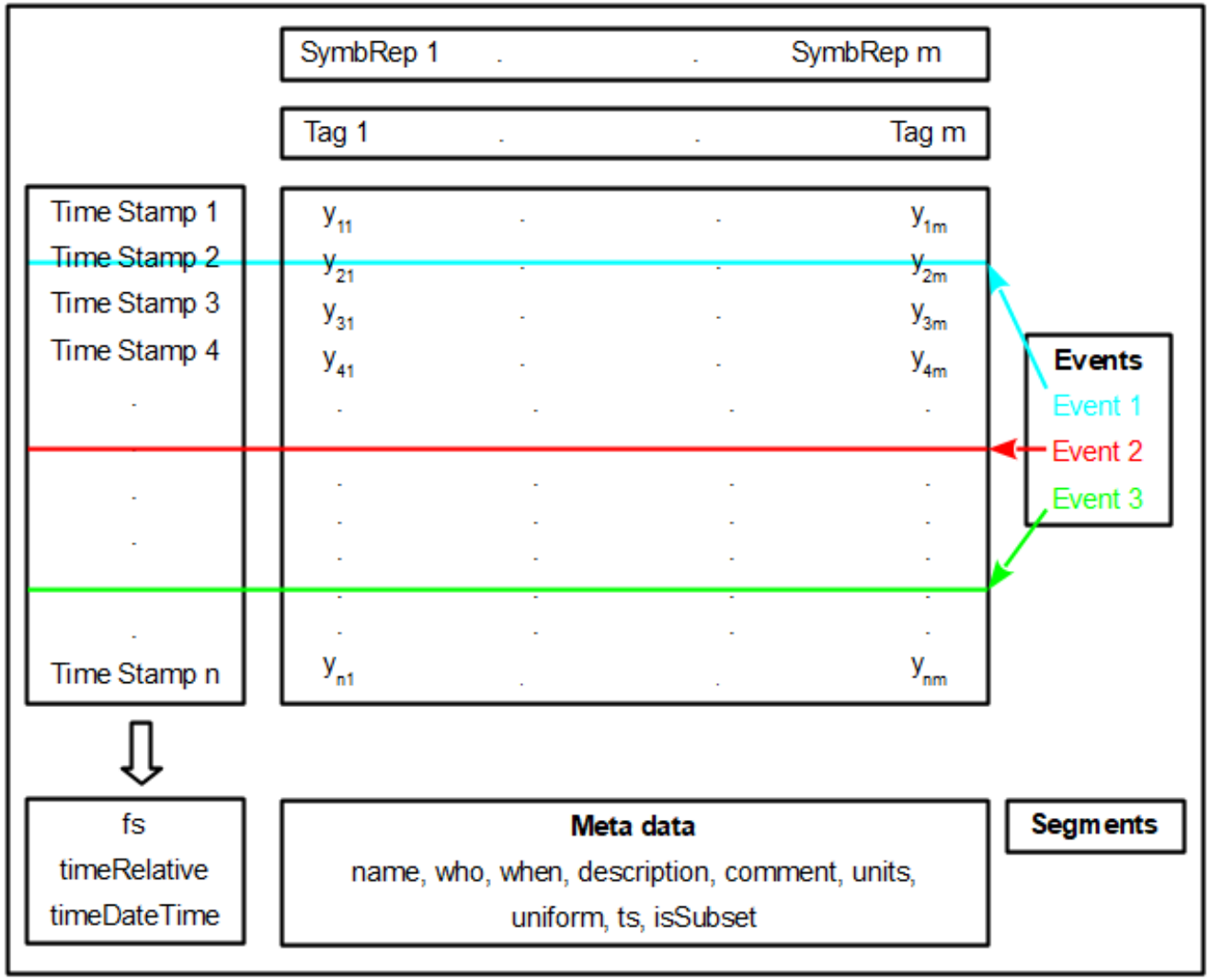
Overview

- What is the mdts-Object?
  - A ‘container’ for time series data (multi-dimensional-time-series), as emanate from systems the CoA is analysing
- What is the benefit of mdts-Objects?
  - Delivers functionality often used
  - Common code base (GitLab)
  - Should be improved from everybody
- Where do I find the mdts-Toolbox
  - Web-Url: https://gitlab.ia.unileoben.ac.at/tsdev/mdtstoolbox
  - Git-Link: git@gitlab.ia.unileoben.ac.at:tsdev/mdtstoolbox.git
- I want to know more:
  - Read the documentation – mdtstoolbox/Documentation/mdtsToolboxDocu.pdf
  - Ask me

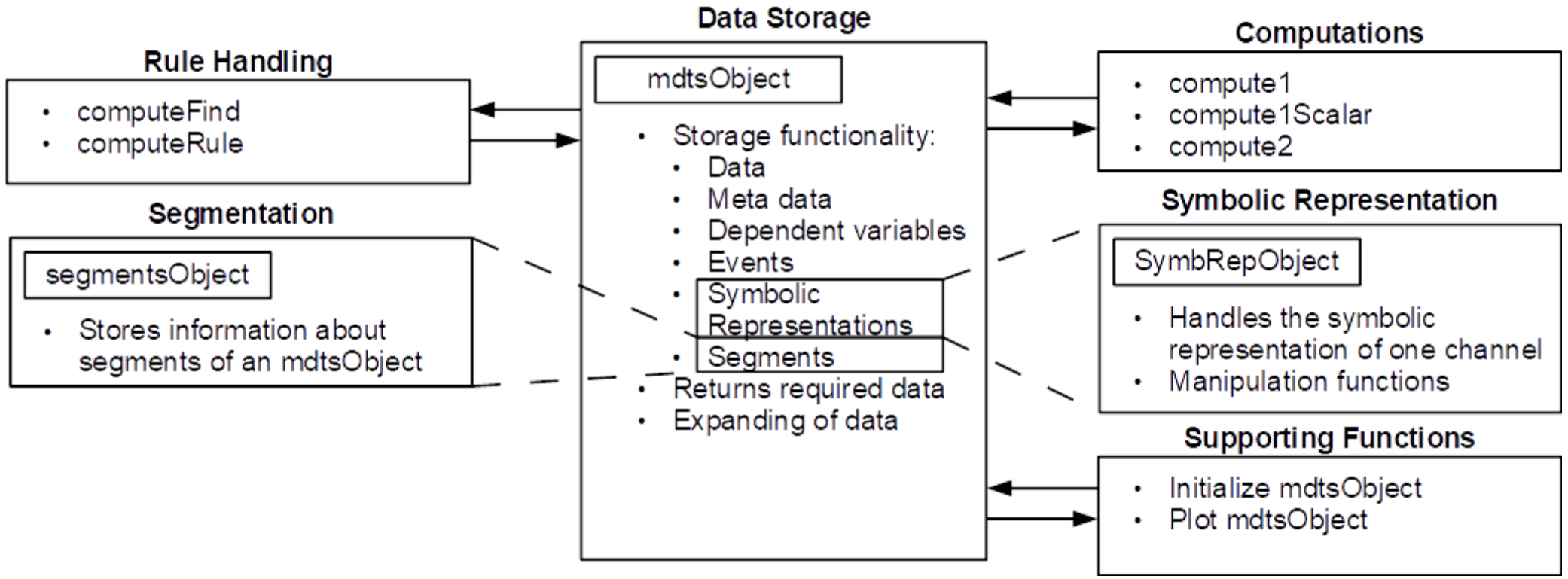
Preliminaries

- Installed Git
- GitLab account
- ssh-key added to your GitLab account
- Download all necessary toolboxes (Web-Urls)
  - Mdtstoolbox : https://gitlab.ia.unileoben.ac.at/tsdev/mdtstoolbox
  - IAToolboxes/figureManager: https://gitlab.ia.unileoben.ac.at/toolboxes/figureManager
  - IAToolboxes/DOPbox: https://gitlab.ia.unileoben.ac.at/toolboxes/DOPbox
  - IAToolboxes/general: https://gitlab.ia.unileoben.ac.at/toolboxes/general
  - IAToolboxes/graphics: https://gitlab.ia.unileoben.ac.at/toolboxes/graphics
- Add Toolboxes to path

Basic Structure (1)

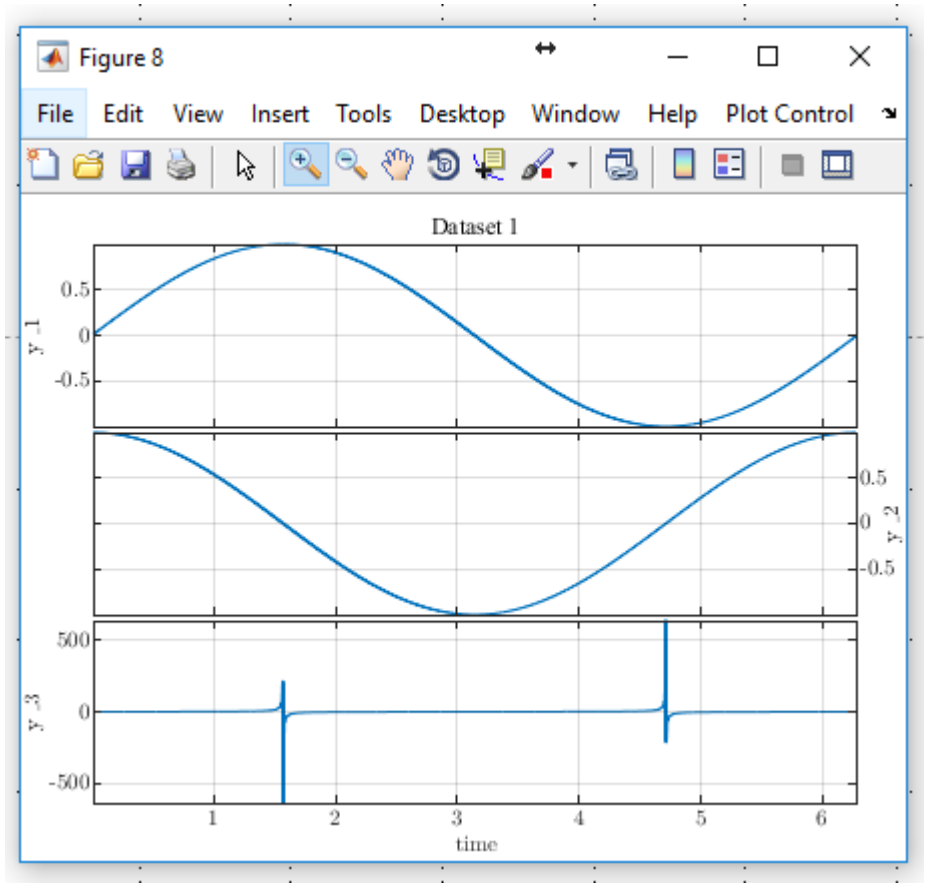


Basic Structure (2)



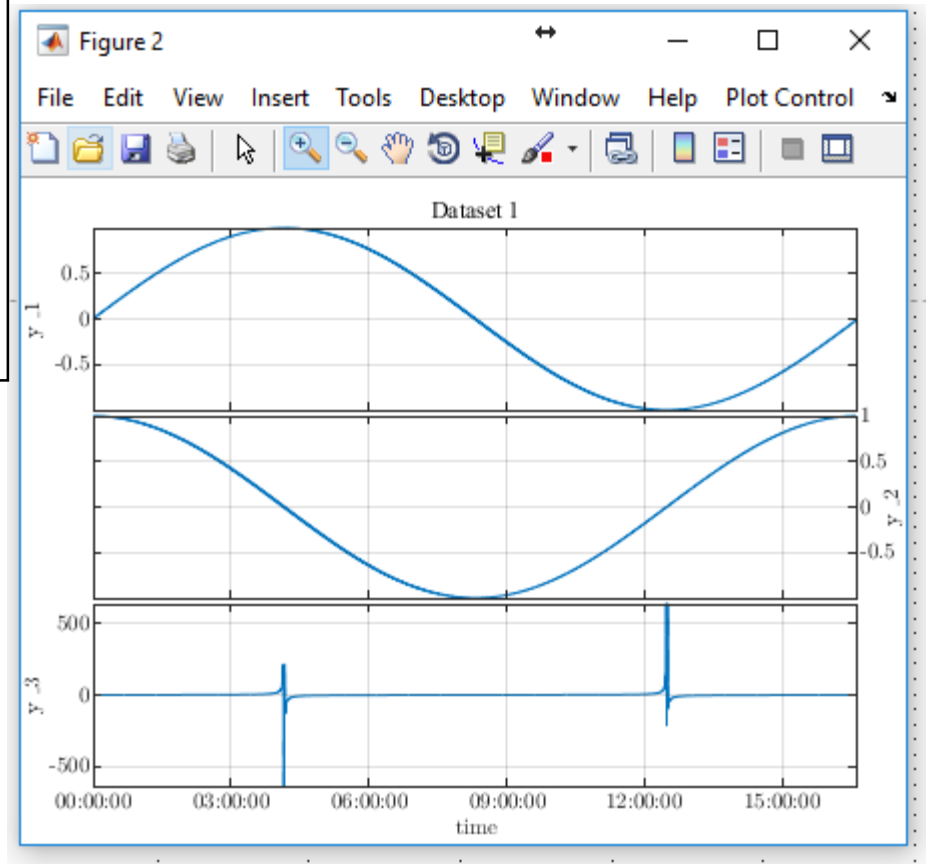
Initialize mdtsObject – numeric x-values

```
%% Initialize mdtsObject - numeric values
n=1000;
%generate dataset
x = linspace(0,2*pi,n)';
y1 = sin(x);
y2 = cos(x);
y3 = tan(x);
D = [y1,y2, y3]; %concatenate data to matrix
tags = {'y_1', 'y_2', 'y_3'} %define the tag names (channel names)
name = 'Dataset 1';
% initialize mdtsObject
mymdtsObject = mdtsObject(x,D, tags, 'name', name);
plotmdtsObject(mymdtsObject);    % plot the object
```



Initialize mdtsObject – Durations

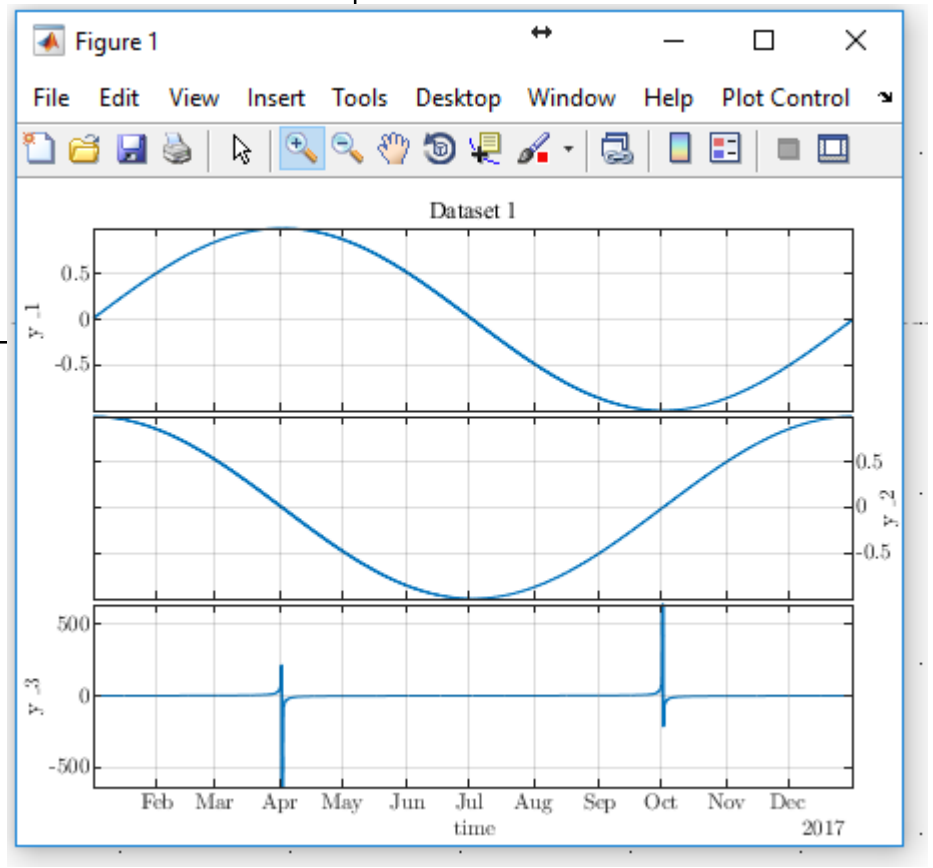
```
%% Initialize mdtsObject - Durations
n=1000;
%generate dataset
x = minutes(1).*(1:n)';
xy = linspace(0,2*pi,n)';
y1 = sin(xy);
y2 = cos(xy);
y3 = tan(xy);
D = [y1,y2, y3]; %concatenate data to matrix
tags = {'y_1', 'y_2', 'y_3'} %define the tag names (channel names)
name = 'Dataset 1';
% initialize mdtsObject
mymdtsObject = mdtsObject(x,D, tags, 'name', name);
plotmdtsObject(mymdtsObject);    % plot the object
```



© 2018, Chair of Automation, University of Leoben
automation.unileoben.ac.at

Initialize mdtsObject – absolute datetime x-values

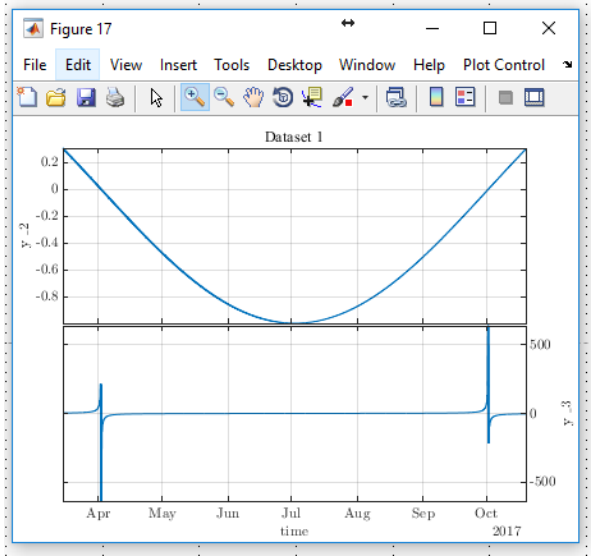
```
%% Initialize mdtsObject
n=1000;
%generate dataset
x = linspace(datetime(2017,1,1), datetime(2018,1,1),n)'; %datetime - xvalues
xy = linspace(0,2*pi,n)';
y1 = sin(xy);
y2 = cos(xy);
y3 = tan(xy);
D = [y1,y2, y3]; %concatenate data to matrix
tags = {'y_1', 'y_2', 'y_3'} %define the tag names (channel names)

mymdtsObject = mdtsObject(x,D, tags); % initialize mdtsObject
plotmdtsObject(mymdtsObject);    % plot the object
```



Extract a subset from mdtsObject – return mdtsObject

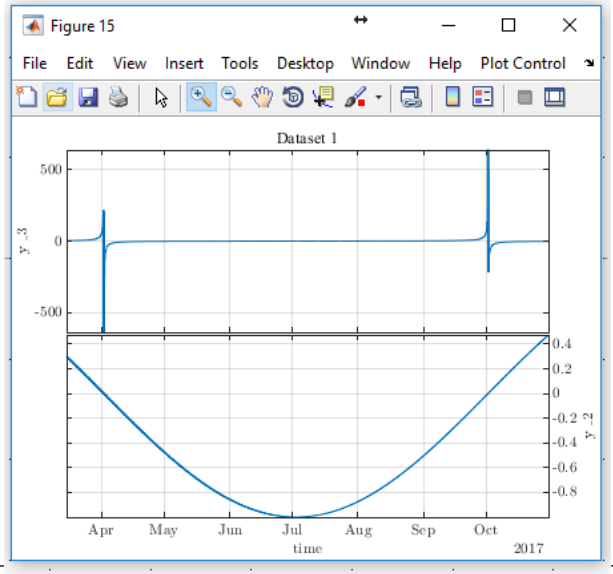
- Direct Indexing
- Maybe use Subfunctions of mdtsObject
  - getInveralIndices
  - getTagIndices

```
%slice
timeInds = (200:800);
tagInds = [2,3];
mymdtsObjectSnippet = mymdtsObject(timeInds,tagInds);

%plot
plotmdtsObject(mymdtsObjectSnippet);    % plot the object
```



- Using functions

```
%slice
timeSnippet = [datetime(2017,3,15), datetime(2017,10,31)];
tagsSnippet = {'y_3', 'y_2'};
mymdtsObjectSnippet = mymdtsObject.getData(tagsSnippet, timeSnippet);

%plot
plotmdtsObject(mymdtsObjectSnippet);    % plot the object
```



Extract data from mdtsObject – return data-Matrix

- Direct Indexing of ‘data’
- Maybe use Subfunctions of mdtsObject
  - getInveralIndices
  - getTagIndices

```
%slice
timeInds = (200:800);
tagInds = [2,3];
dataExtracted = mymdtsObject.data(timeInds,tagInds);
```

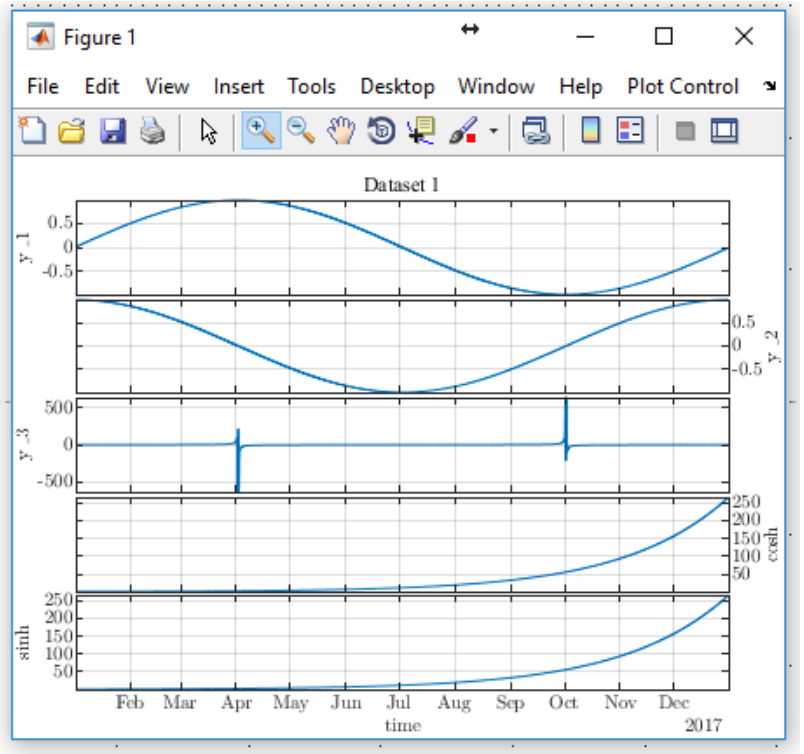
```
>> dataExtracted

dataExtracted =

    0.3138    3.0258
    0.3078    3.0909
    0.3018    3.1586
    0.2958    3.2290
    0.2898    3.3024
    0.2838    3.3789
    0.2778    3.4587
    0.2717    3.5420
    0.2656    3.6291
    0.2596    3.7203
    0.2535    3.8159
    0.2474    3.9162
    0.2413    4.0216
    0.2352    4.1324
```

- Using functions

```
%slice
timeSnippet = [datetime(2017,3,15), datetime(2017,10,31)];
tagsSnippet = {'y_3', 'y_2'};
mymdtsObjectSnippet = mymdtsObject.getRawData(tagsSnippet, timeSnippet);
```

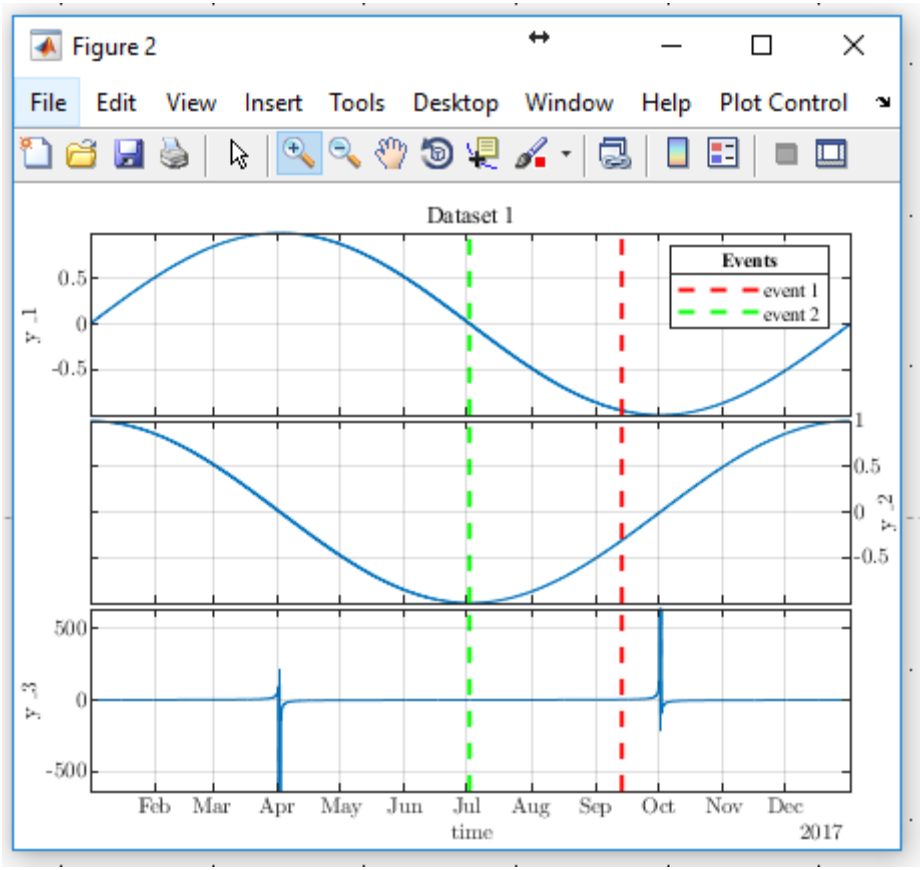
Add data to mdtsObject – ‘expandDataSet(data,tags)

```
%add data to mdtsObject
y_cosh = cosh(xy);
y_sinh = sinh(xy);
data_expand = [y_cosh, y_sinh];
tags_expand = {'cosh', 'sinh'};

mymdtsObject.expandDataSet(data_expand, tags_expand);
plotmdtsObject(mymdtsObject);
```



Add event to mdtsObject – ‘addEvent(eventID, timep,duration)’

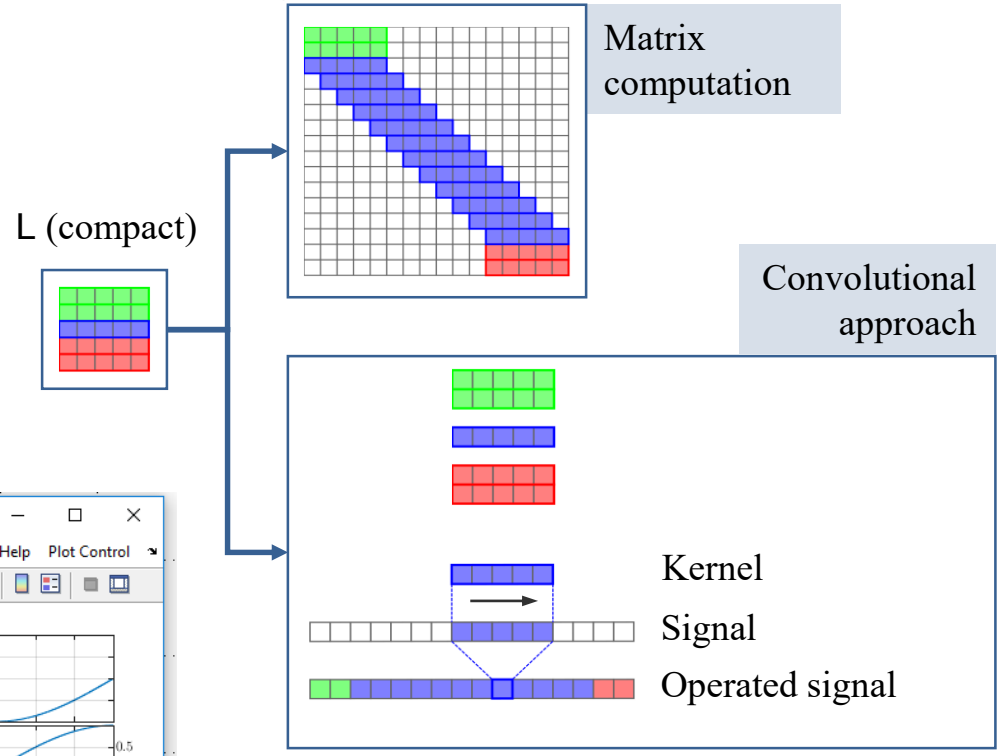
```
event1time = x(700);
event1ID = 'event 1';
event1duration = seconds(5);

event2time = x(500);
event2ID = 'event 2';
event2duration = seconds(3);

mymdtsObject.addEvent(event1ID,event1time,event1duration);
mymdtsObject.addEvent(event2ID,event2time,event2duration);
plotmdtsObject(mymdtsObject);
```



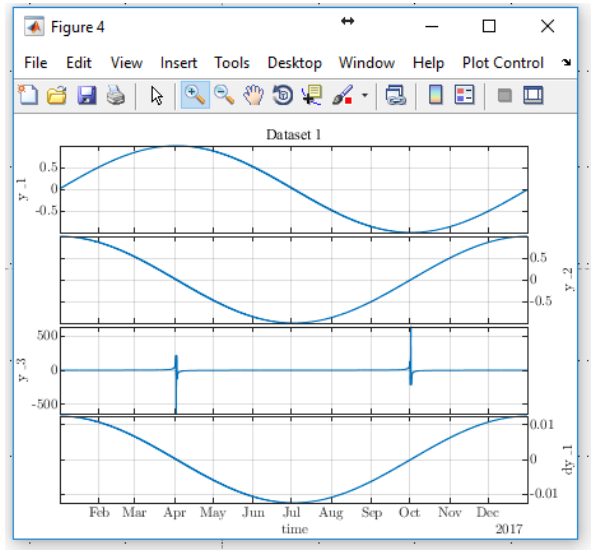
- Only accepts timepoints from x-axis?? Maybe allow every timepoint

Perform calculations – compute1(matrix,input)

- Performs a calculation on one data-vector (one channel) x
  - matrix: matrix to perform local computation
  - Input: either a vector of data or a struct
    - Inputstruct.tag: string containing the tagname of channel in mdtsObject to apply the computation on
    - Inputstruct.object: the mdtsObject to apply the computation



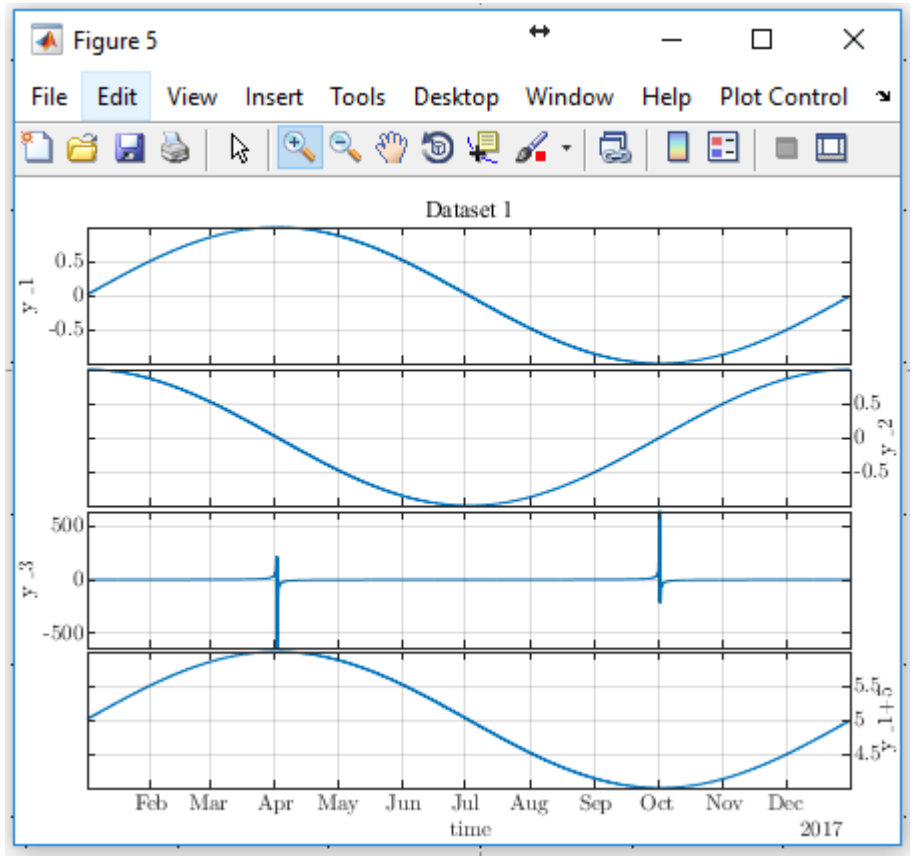

```
[B,dB] = dop(ls);
DMat = dB*B';

input1.tag = 'y_1';
input1.object= mymdtsObject;

dy_1 = compute1(DMat, input1);
mymdtsObject.expandDataSet(dy_1, 'dy_1');

plotmdtsObject(mymdtsObject);
```



Perform calculations – compute1Scalar(operator,scalar,input)

- Performs elementwise operations with the given scalar:
  - operator: +, - , / , * , @funHandle
  - input: vector or struct containing tag and object
    - Struct.tag = ‘tag’
    - Struct.object = ‘mymdtsObject’
  - Scalar: a scalar value

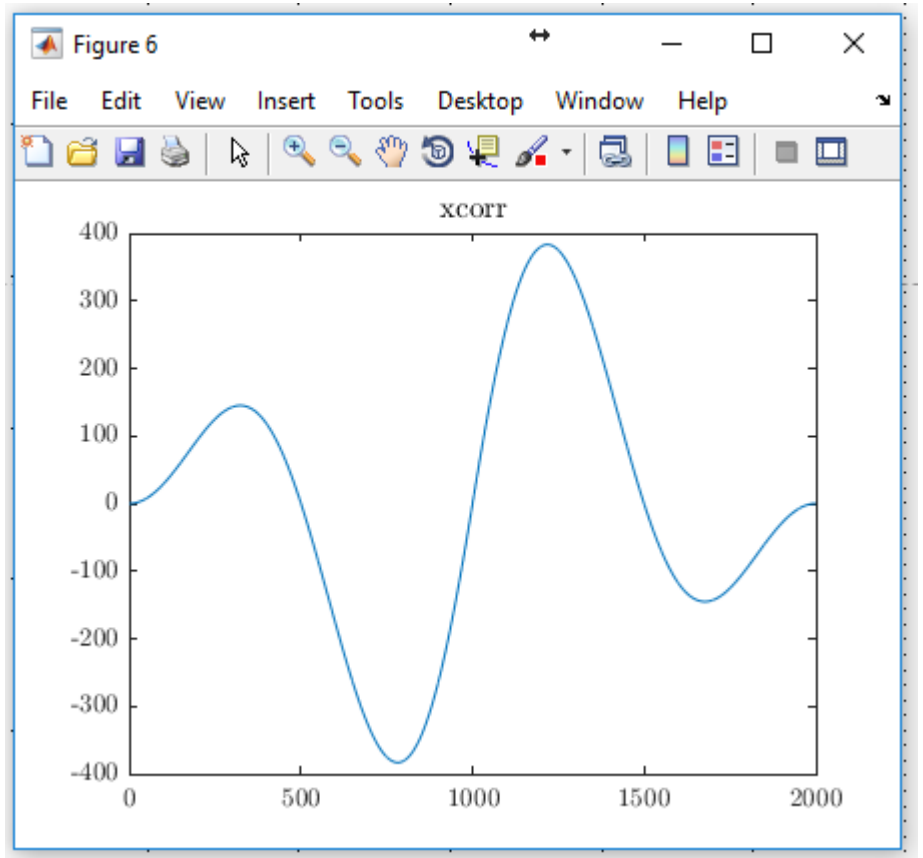
```
input1.tag = 'y_1';
input1.object= mymdtsObject;
operator = '+';
scalarVal = 5;

y_15 = compute1Scalar(operator, scalarVal,input1);
mymdtsObject.expandDataSet(y_15, 'y_1+5');

plotmdtsObject(mymdtsObject);
```



Perform Calculations – compute2(operator, input1, input2)

- Performs operations on two input vectors/signals
  - operator: *, /, +, -, dot, outer, xcorr, @funHandle
  - Input1/2: vector or struct containing tag and object
    - Struct.tag = ‘tag’;
    - Struct.object = ‘mymdtsObject’;

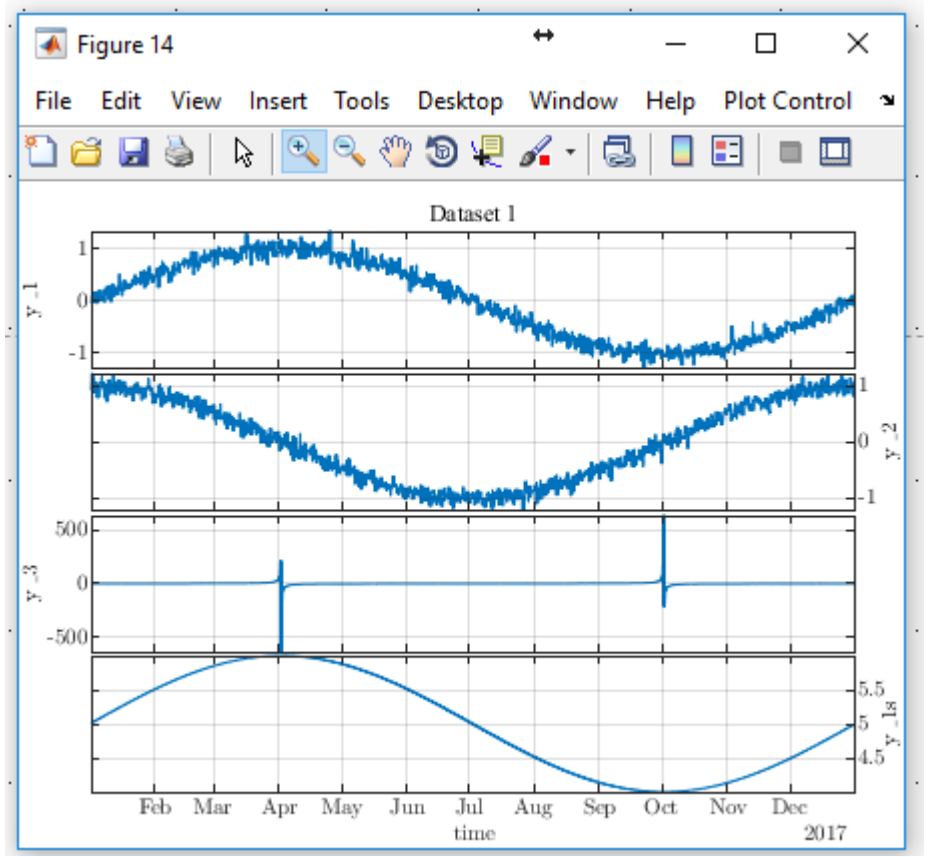
```
input1.tag = 'y_1';
input1.object= mymdtsObject;
input2.tag = 'y_2';
input2.object= mymdtsObject;

operator = 'xcorr';

xcorry_12 = compute2(operator, input1, input2);

figureGen;
plot(xcorry_12);
title('xcorr');
```



Perform Calculations – smoothAsConv(input, varargin)

- Performs local smoothing on given input using convolutional approach in combination with dop-Box
  - Input vector or struct containing tag and object
  - Varargin: key-value pairs – ls, noBfs
  - It is assumed, that the x-axis is evenly spaced

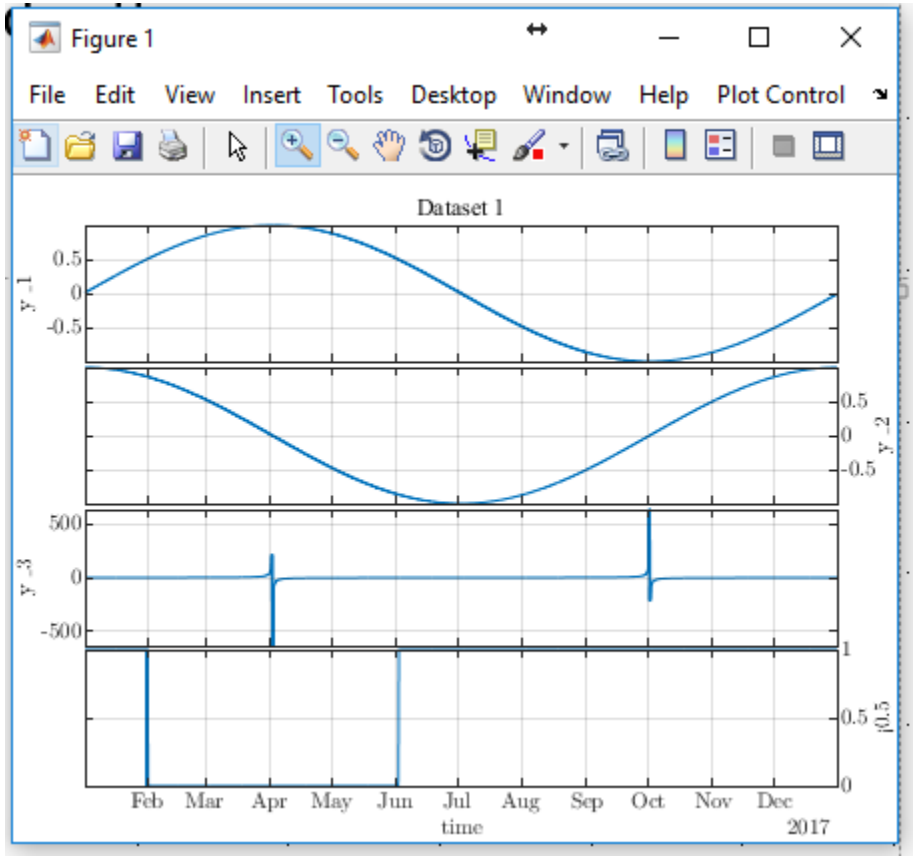
```
input1.tag = 'y_1';
input1.object= mymdtsObject;
y1s = smoothAsConv(input1, 'ls', 11, 'noBfs', 3);
mymdtsObject.expandDataSet(y_15, 'y_1s');
plotmdtsObject(mymdtsObject);
```



Perform Calculations – computeFind(operator, input, value)

- Finds values in the input which meet the given criteria and value
  - Operator: >, <, ==, ~=, >=, <=
  - Input: vector or struct containing tag and object
  - Value: a scalar to which the input should be compared with

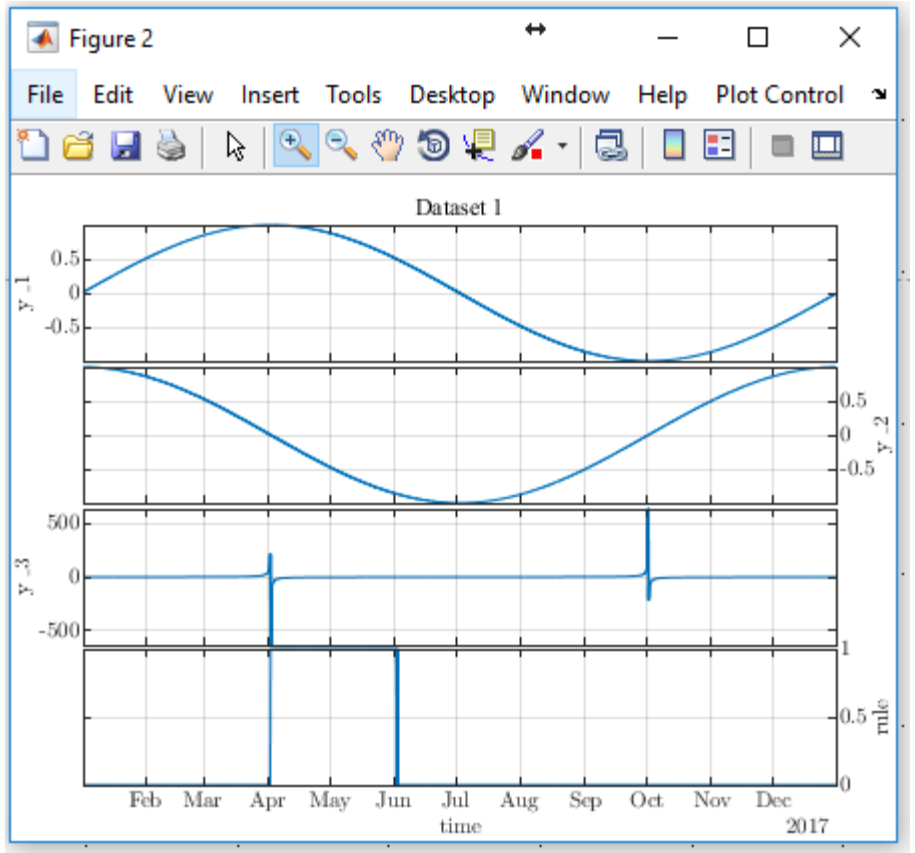
```
input1.tag = 'y_1';
input1.object= mymdtsObject;

valueComp = 0.5;

ruleRes = computeFind('<', input1, valueComp);
mymdtsObject.expandDataSet(ruleRes, '<0.5');

plotmdtsObject(mymdtsObject);
```



Perform Calculations – computeRule(operator, arr1, arr2)

- Used to perform logical operations on two logical arrays
  - Operator: &, |
  - Arr1/2: logical arrays for which the computation should be performed

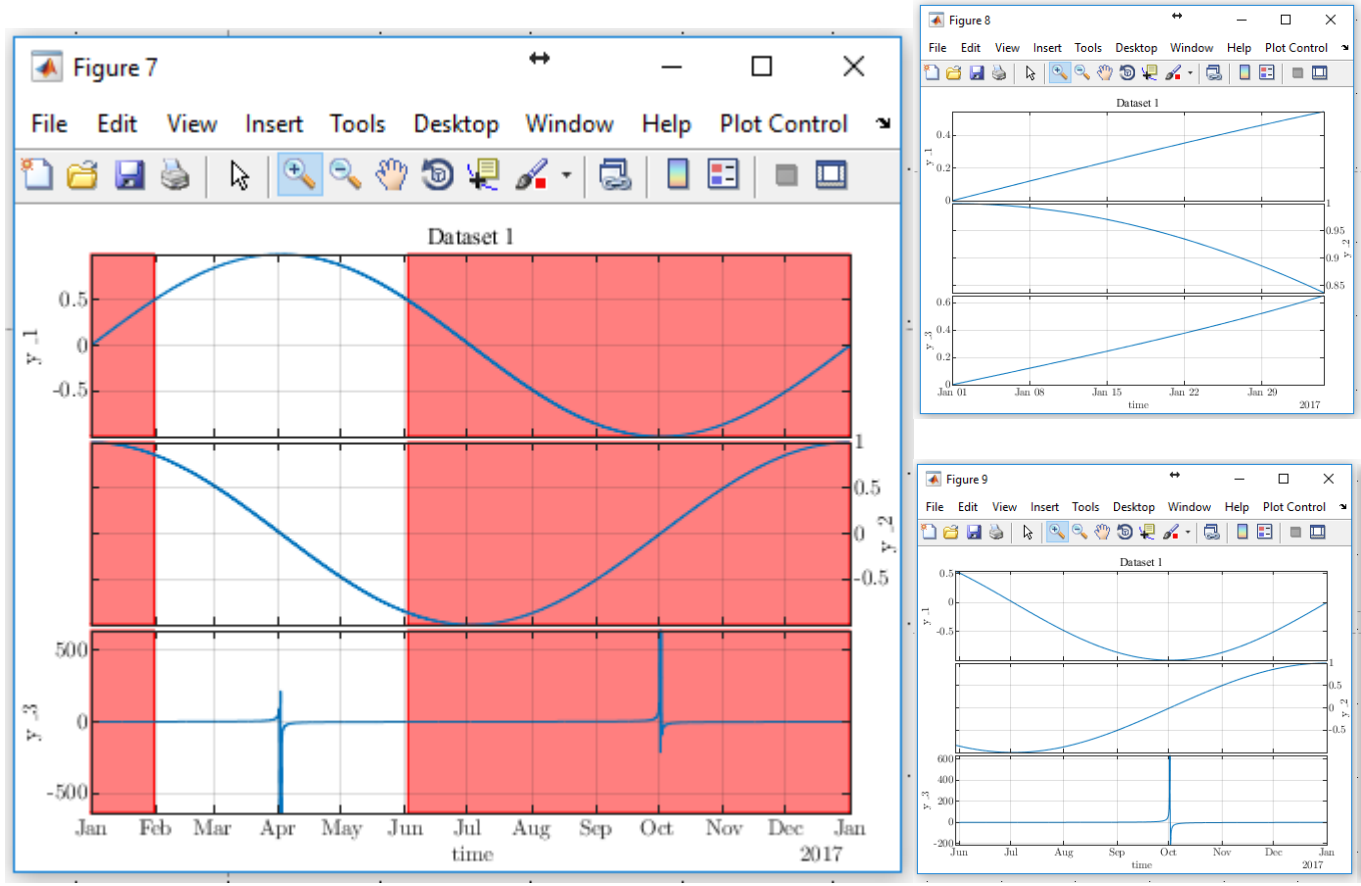
```
input1.tag = 'y_1';
input1.object= mymdtsObject;
input2.tag = 'y_2';
input2.object=mymdtsObject;

valueComp1 = 0.5;
valueComp2 = 0;

findRes1 = computeFind('>', input1, valueComp1);
findRes2 = computeFind('<', input2, valueComp2);

ruleOperator = '&';
ruleRes = computeRule(ruleOperator, findRes1,
findRes2);

mymdtsObject.expandDataSet(ruleRes, 'rule');

plotmdtsObject(mymdtsObject);
```



Segments (under development)

- Goal: define segments as own objects – can be used to extract data from timeseries objects

```
input1.tag = 'y_1';
input1.object= mymdtsObject;

valueComp1 = 0.5;


findRes1 = computeFind('<', input1, valueComp1);

nRow = length(mymdtsObject.time);
mySeg1 = segmentsObject(nRow);
segTag = 'segment1';
mySeg1 =mySeg1.addSegmentVector(segTag, findRes1);
mymdtsObject.addSegments(mySeg1);

plotSegments(mymdtsObject, segTag);

segsMdtsObjects = extractSegments(mymdtsObject,
segTag);
for i=1:length(segsMdtsObjects)
    plotmdtsObject(segsMdtsObjects{i});
end
```



Symbolic Analyisis