Course unit title:
Basics of Information Systems
Course unit code:
NIRIA1SEND

Matlab basics III.

Dr. habil. Levente Kovács
associate professor

Obuda University,
John von Neumann Faculty of Informatics,
Department of Information Systems,
Physiological Controls Group

24.09.2013.

Review – last lecture

- Create 2-D line plot
- LineSpec
- Plot multiple lines
- hold
- Add title and axis labels
- Subplot
- Create 3-D line plot
- Create 3-D shaded surface plot
- Reading sound files
- Reading image from graphics file
- Loading Data

Contents

The Matlab Basic Fitting GUI

The MATLAB® Basic Fitting GUI allows you to interactively:

- Model data using a spline interpolant, a shape-preserving interpolant, or a polynomial up to the tenth degree
- Plot one or more fits together with data
- Plot the residuals of the fits
- Compute model coefficients
- Compute the norm of the residuals (a statistic you can use to analyze how well a model fits your data)
- Use the model to interpolate or extrapolate outside of the data
- Save coefficients and computed values to the MATLAB workspace for use outside of the GUI
- Generate MATLAB code to recompute fits and reproduce plots with new data

Preparing for Basic Fitting

- the Basic Fitting GUI is only available for 2-D plots
  - ✓ for more advanced fitting and regression analysis →Curve Fitting Toolbox™ documentation and the Statistics Toolbox™ documentation
- the Basic Fitting GUI sorts the data in ascending order before fitting
  - ✓ if the dataset is large and the values are not sorted in ascending order → it will take longer
- speed up the Basic Fitting GUI by first sorting your data
  - ✓ create sorted vectors x_sorted and y_sorted from data vectors x and y → `sort` function

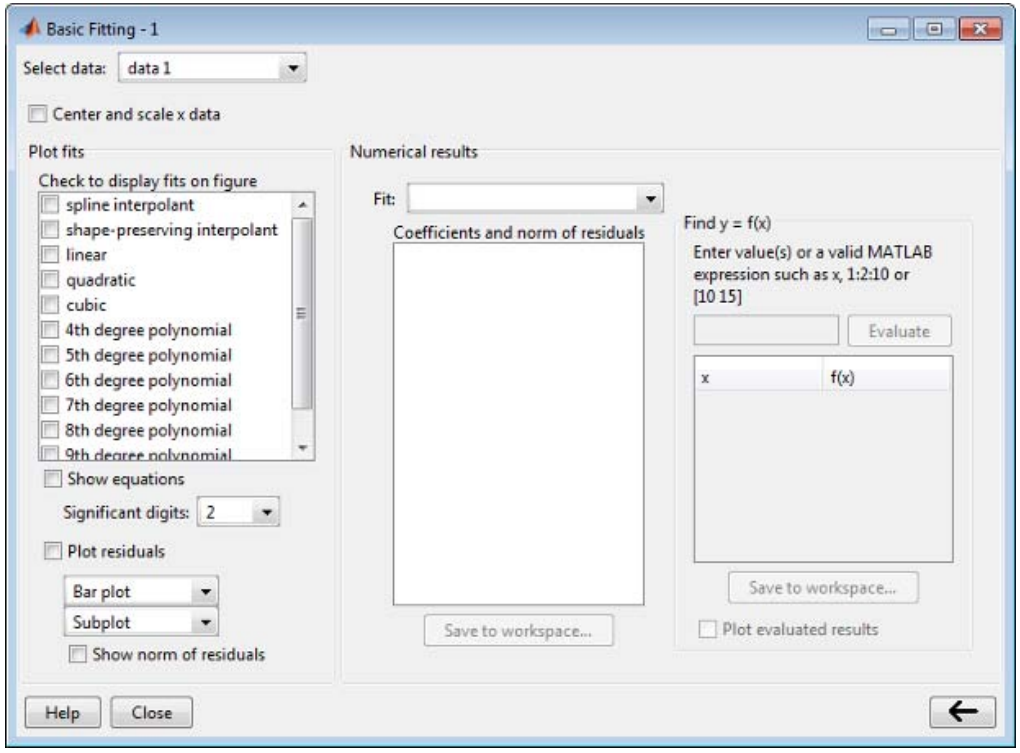
```
[x_sorted, i] = sort(x);
y_sorted = y(i);
```

Opening the Basic Fitting GUI

- to use the Basic Fitting GUI, you must first plot your data in a figure window
- open the Basic Fitting GUI, select **Plot > Tools > Basic Fitting**
- when you fully expand it by double clicking the arrow button in the lower right corner, the window displays three panels

✓ use these panels to:

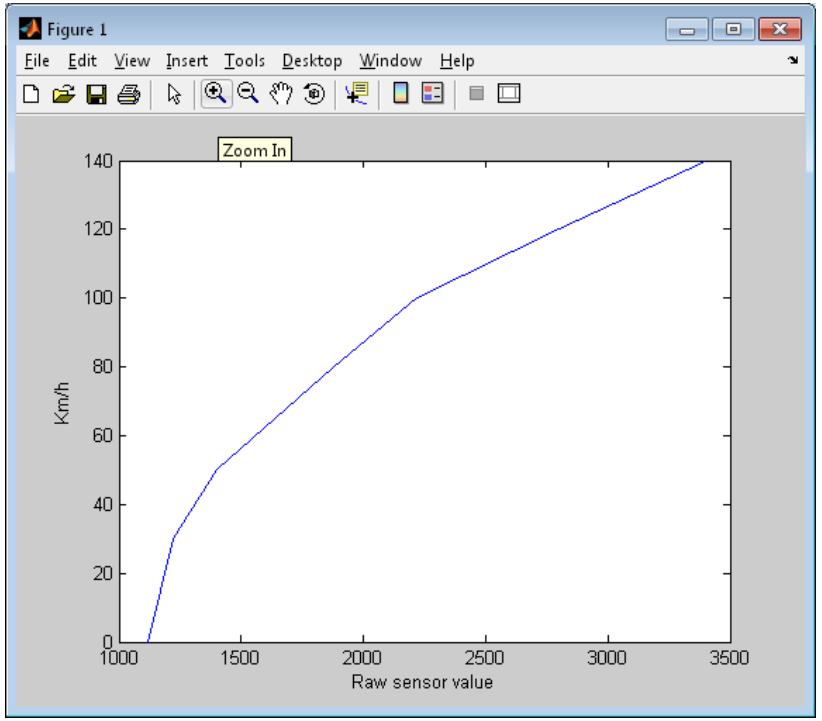
1. Select a model and plotting options
2. Examine and export model coefficients and norms of residuals
3. Examine and export interpolated and extrapolated values.



Using the Basic Fitting GUI – example

1. Plot the data in a figure window

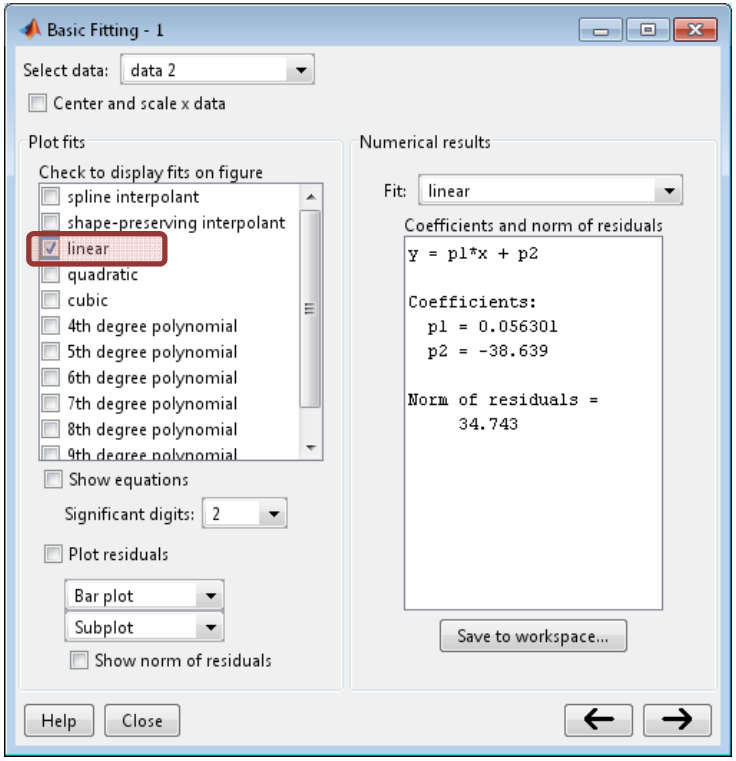
```
airspeed=[
0   1120    ;
30  1220    ;
50  1400    ;
80  1880    ;
100 2220    ;
120 2800    ;
140 3400    ;
]

plot(airspeed(:,2), airspeed(:,1))
ylabel('Km/h')
xlabel('Raw sensor value')
```

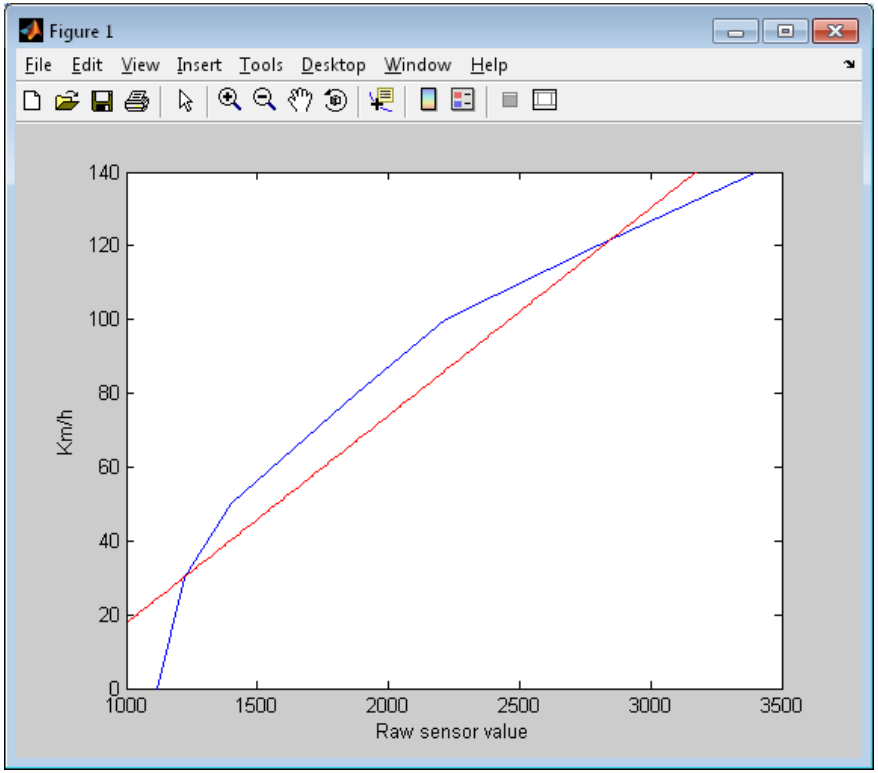


Using the Basic Fitting GUI – example

2. Open the Basic Fitting GUI

✓ linear fitting

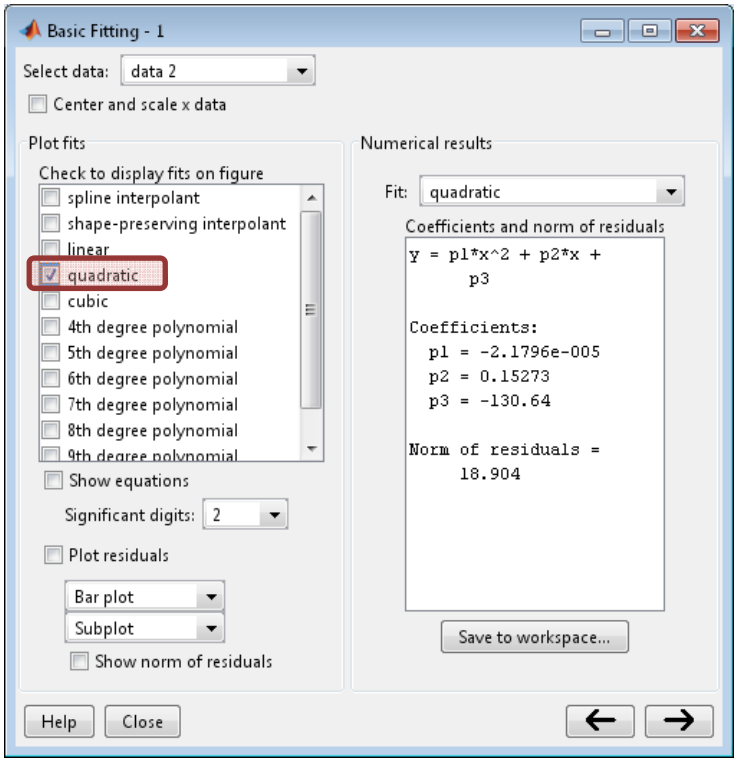


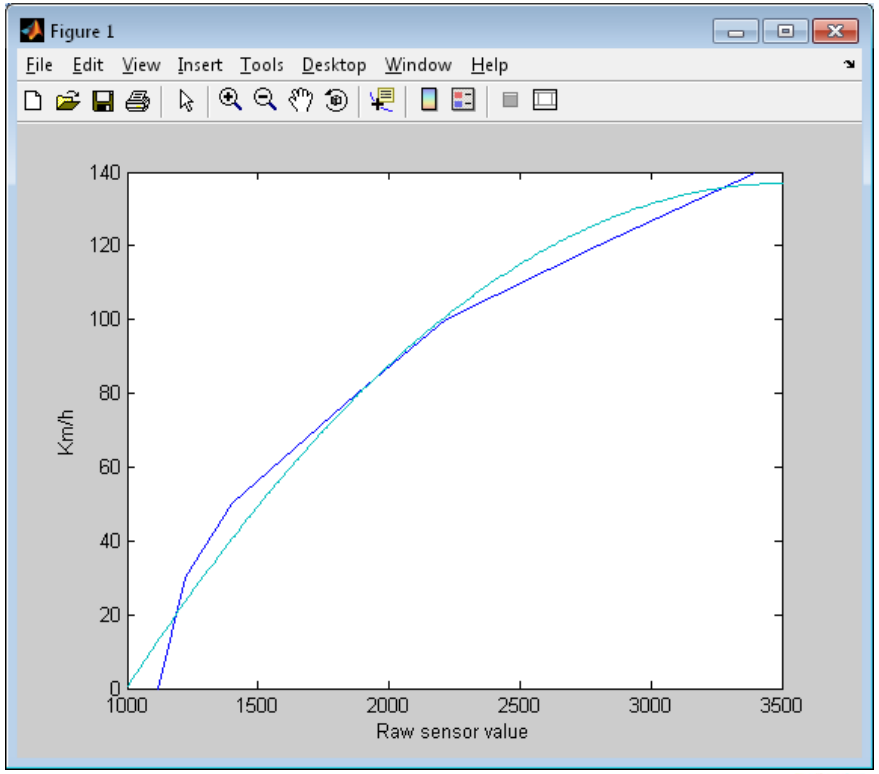


Using the Basic Fitting GUI – example

2. Open the Basic Fitting GUI

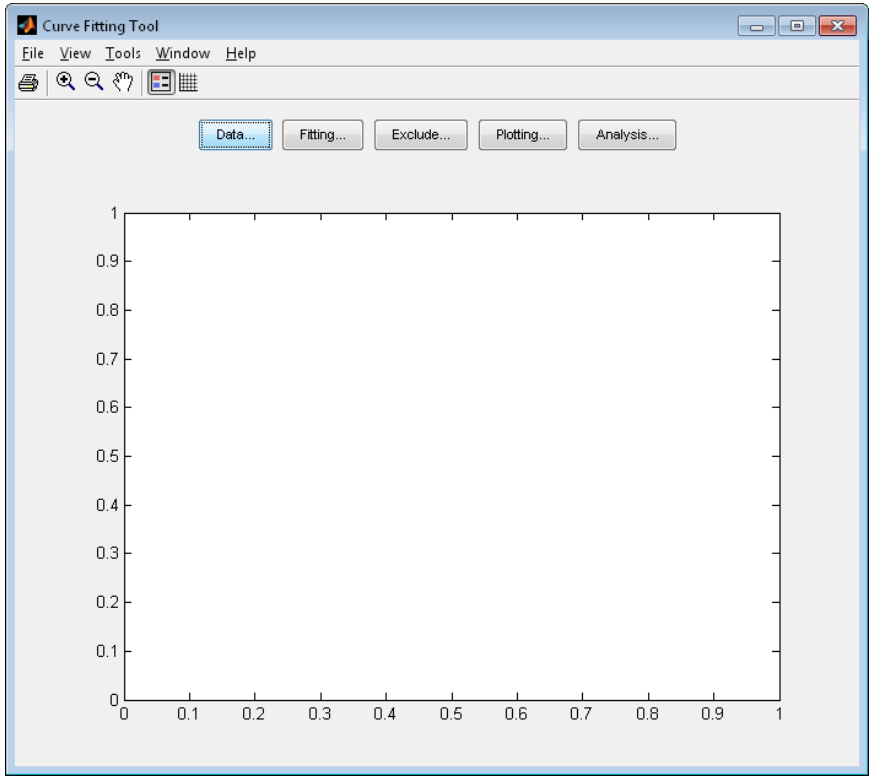
✓ quadratic fitting





Curve Fitting Tool

```
cftool
```



- a flexible interface to **interactively fit curves and surfaces** to data and view plots
  - ✓ create, plot, and compare multiple fits
  - ✓ use linear or nonlinear regression, interpolation, local smoothing regression, or custom equations
  - ✓ view goodness-of-fit statistics, display confidence intervals and residuals, remove outliers and assess fits with validation data
  - ✓ automatically generates code for fitting and plotting surfaces, or export fits to workspace for further analysis

Curve Fitting Tool – example

Declaring variable from the Workspace

```
cftool(x,y)
```

- Creates a curve fit to *x* input and *y* output (*x* and *y* must be numeric, have two or more elements, and have the same number of elements)

Declaring variable with Curve Fitting Tool GUI

cftool > Data

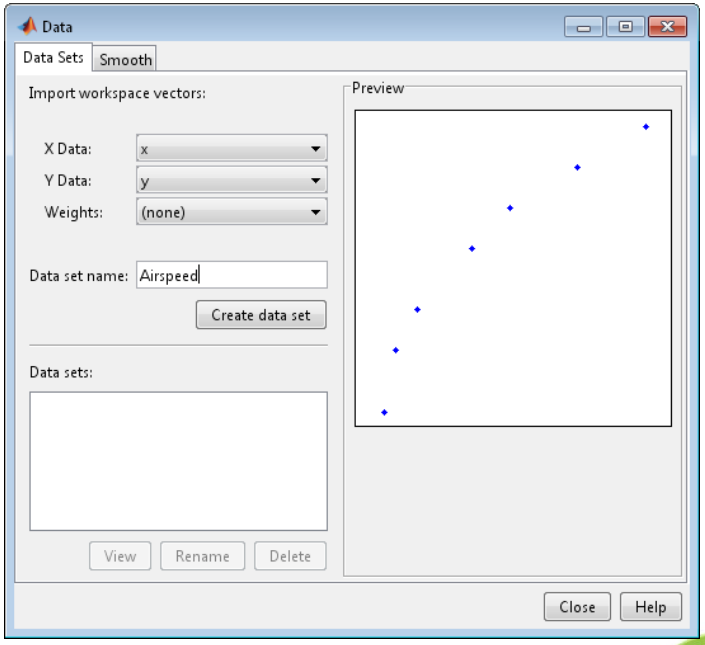
```
x=airspeed(:,2)
y=airspeed(:,1)
cftool
```

X Data: x

Y Data: y

Data set name: Airspeed

-> Create data set



Curve Fitting Tool – example

Curve fitting from the Workspace

```
fitobject = fit(x,y,fitType)
```

- Creates the fit to the data in *x* and *y* with the model specified by *fitType*

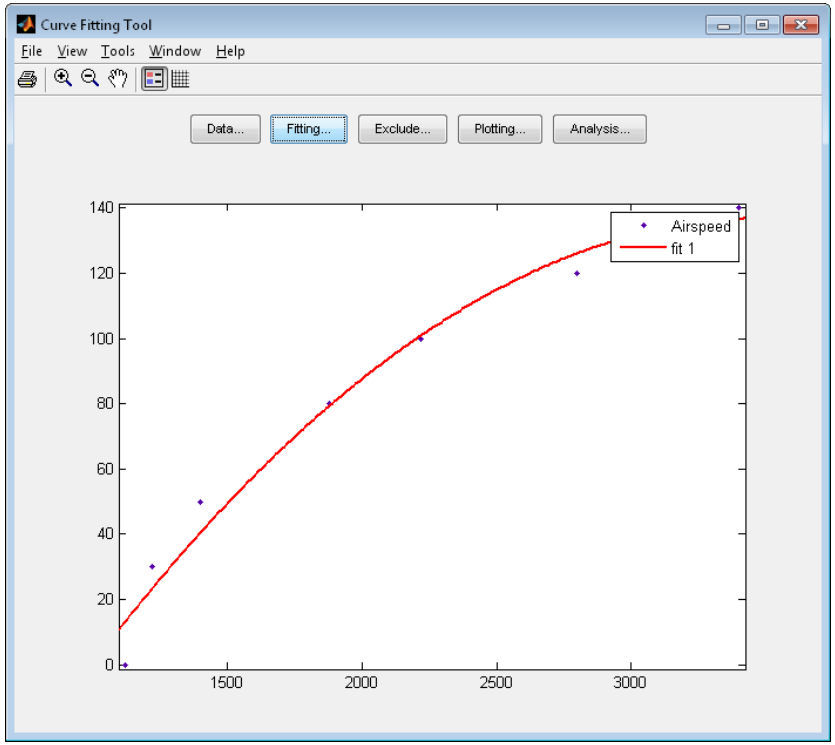
Library Model Name	Description
'poly1'	Linear polynomial curve
'poly11'	Linear polynomial surface
'poly2'	Quadratic polynomial curve
'linearinterp'	Piecewise linear interpolation
'cubicinterp'	Piecewise cubic interpolation
'smoothingspline'	Smoothing spline (curve)
'lowess'	Local linear regression (surface)

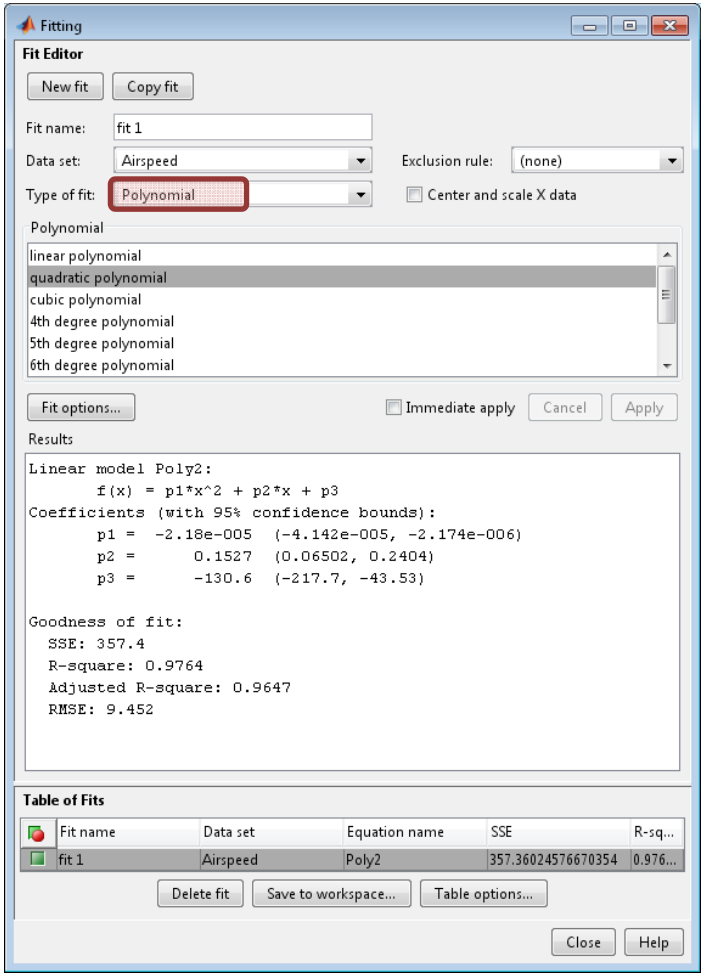
```
f=fit(x,y,'poly2')
plot(f,x,y)
```

Curve Fitting Tool – example

Curve fitting with Curve Fitting Tool GUI

cftool > Fitting

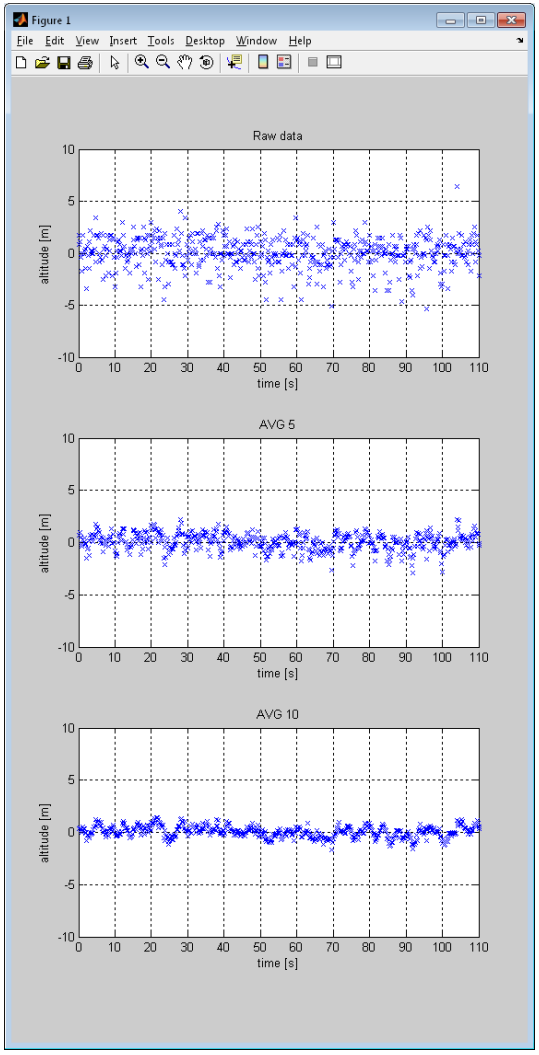




Signal filtering

Signal filtering

1. mean
2. median
3. Sliding Window Filter



mean

```
M = mean(A)
```

• Returns the mean values of the elements along different dimensions of an array.

If *A* is a vector, `mean(A)` returns the mean value of *A*

```
a=[1 3 5 7 9]
a =
     1   3   5   7   9

mean(a)
ans =
     5
```

If *A* is a matrix, `mean(A)` treats the columns of *A* as vectors, returning a row vector of mean values

```
A = [1 2 3;
     3 3 6;
     4 6 8;
     4 7 7];
mean(A)
ans = 3.0000 4.5000 6.0000
```

median

```
M = median(A)
```

- Returns the median value of A

  (Median is the numerical value separating the higher half of a data sample from the lower half.)

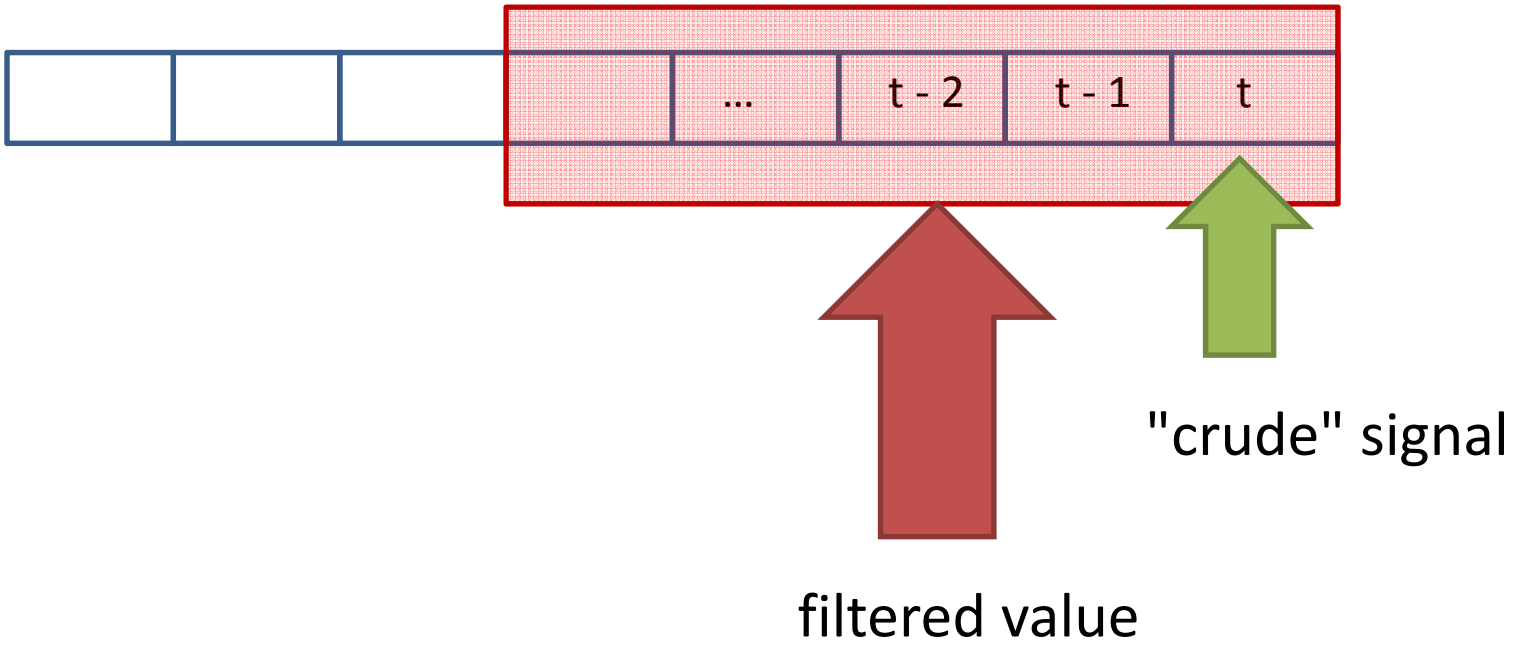
If *A* is a vector, `median(A)` returns the median value of *A*

```
a =
    4   11   23   2    1

median(a)
ans =
     4
```

If *A* is a nonempty matrix, then `median(A)` treats the columns of *A* as vectors and returns a row vector of median values.

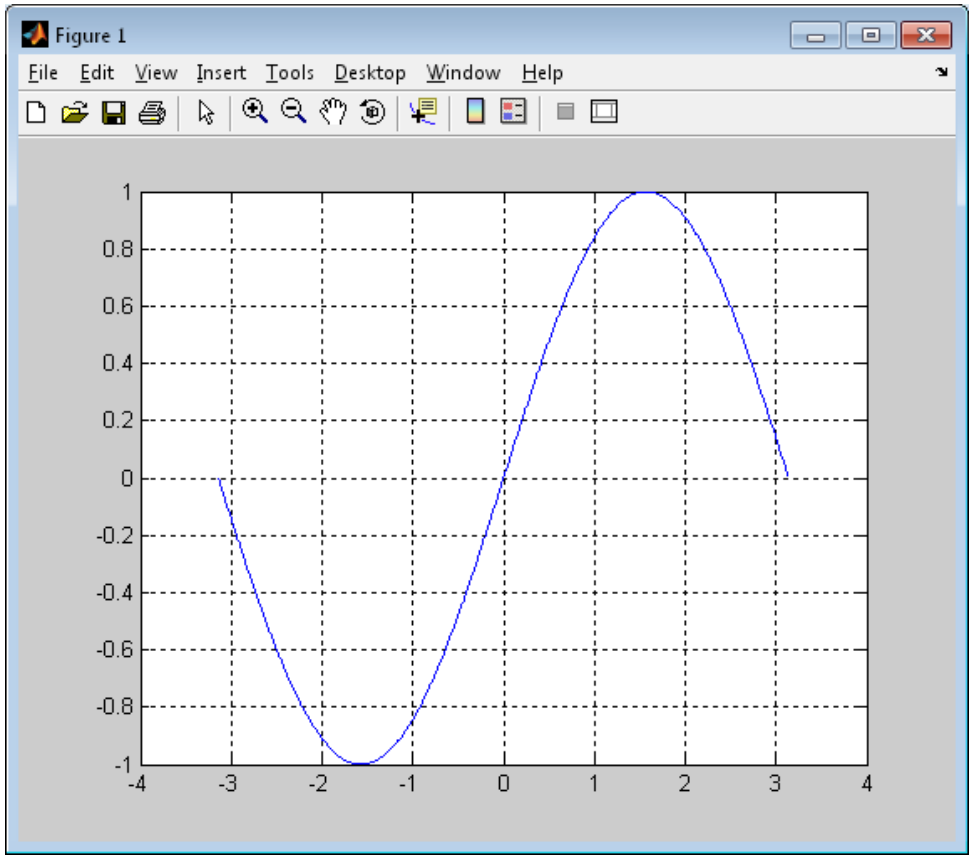
```
A = [0 1 1;
     2 3 2;
     1 3 2;
     4 2 2]
M = median(A)
M = 1.5000 2.5000 2.0000
```

Sliding Window Filter



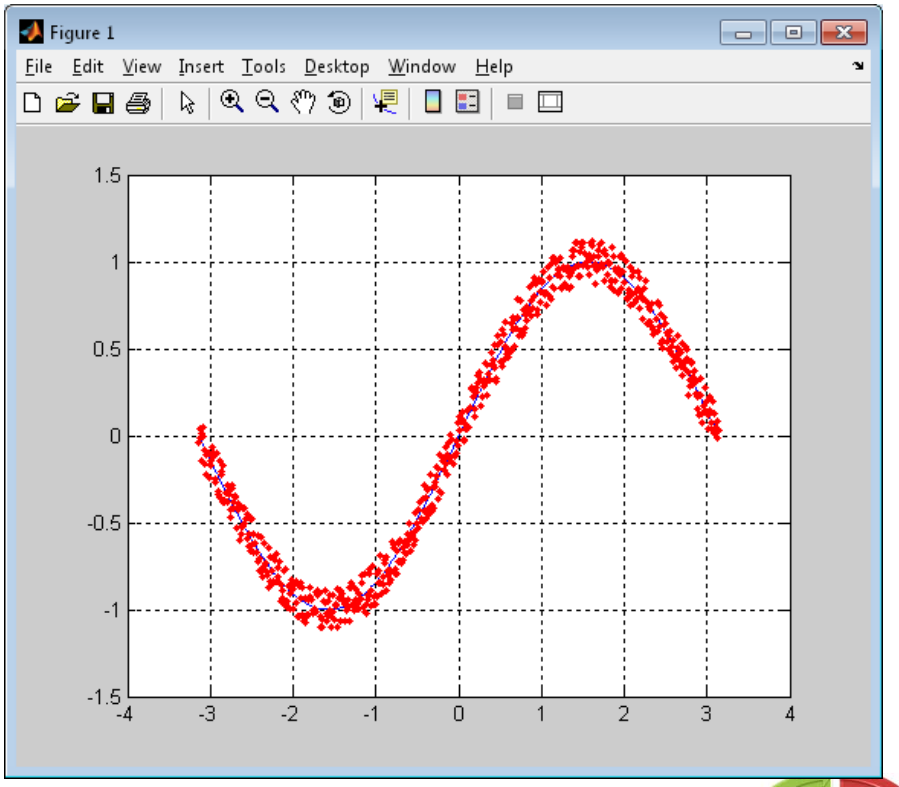
Reference signal

```
x=[-pi:0.01:pi];
y= sin(x);
plot(x,y);
```



Noise

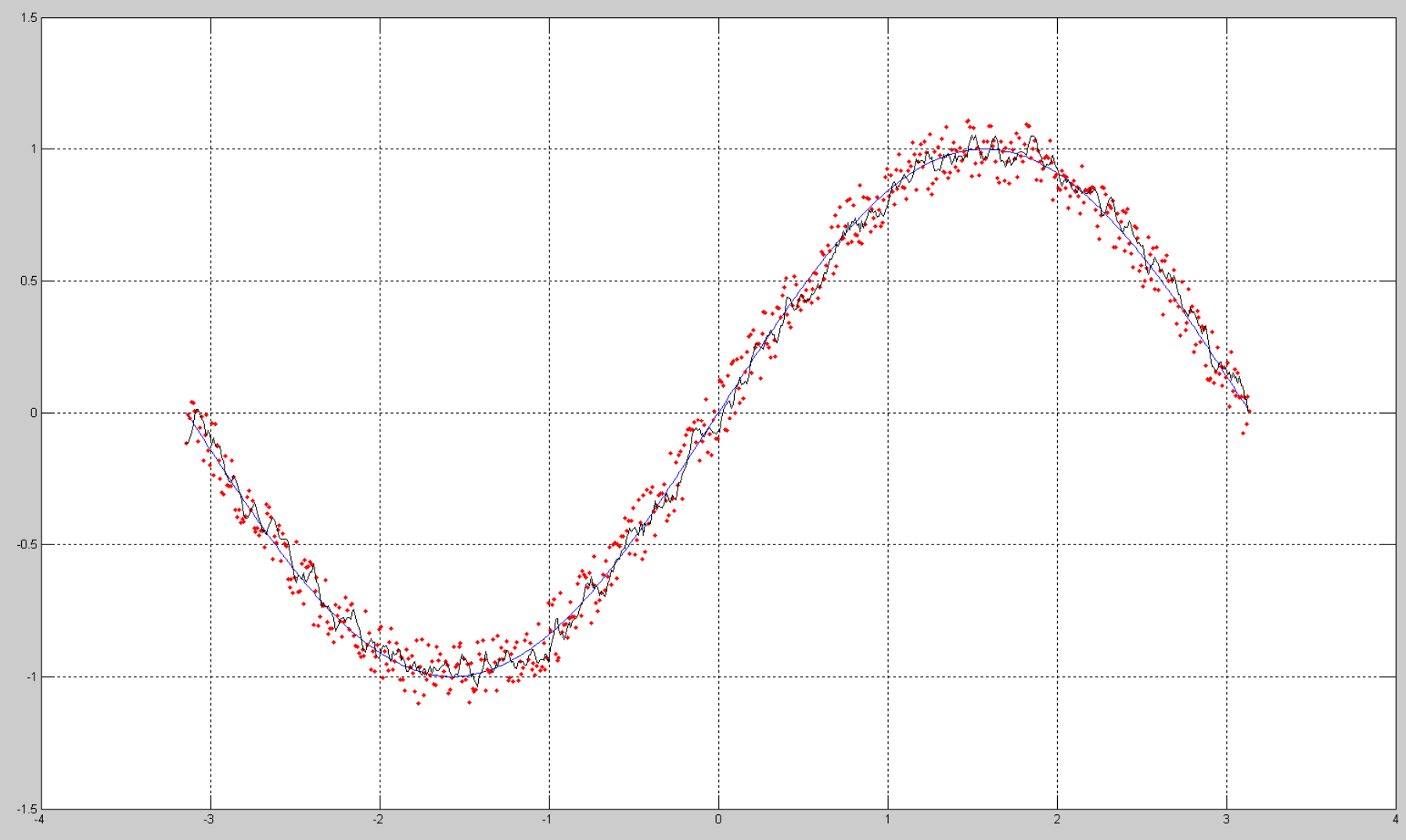
```
y2=0;
for i=1:length(x)
    y2(i)=sin(x(i))+(rand(1)-0.5)/4;
end
hold on;
plot(x,y2,' .r');
```



Sliding Window Filter – example

```
slidewindow=[y2(1) y2(1) y2(1) y2(1) y2(1)
];
y2_filtered=0;
for i=1:length(y2)
    y2_filtered(i)=mean(slidewindow);
    %y2_filtered(i)=median(slidewindow);
    for j=1:length(slidewindow)-1
        slidewindow(j)=slidewindow(j+1);
    end;

    slidewindow(length(slidewindow))=y2(i);
end;

plot(x,y2_filtered,'-k');
```

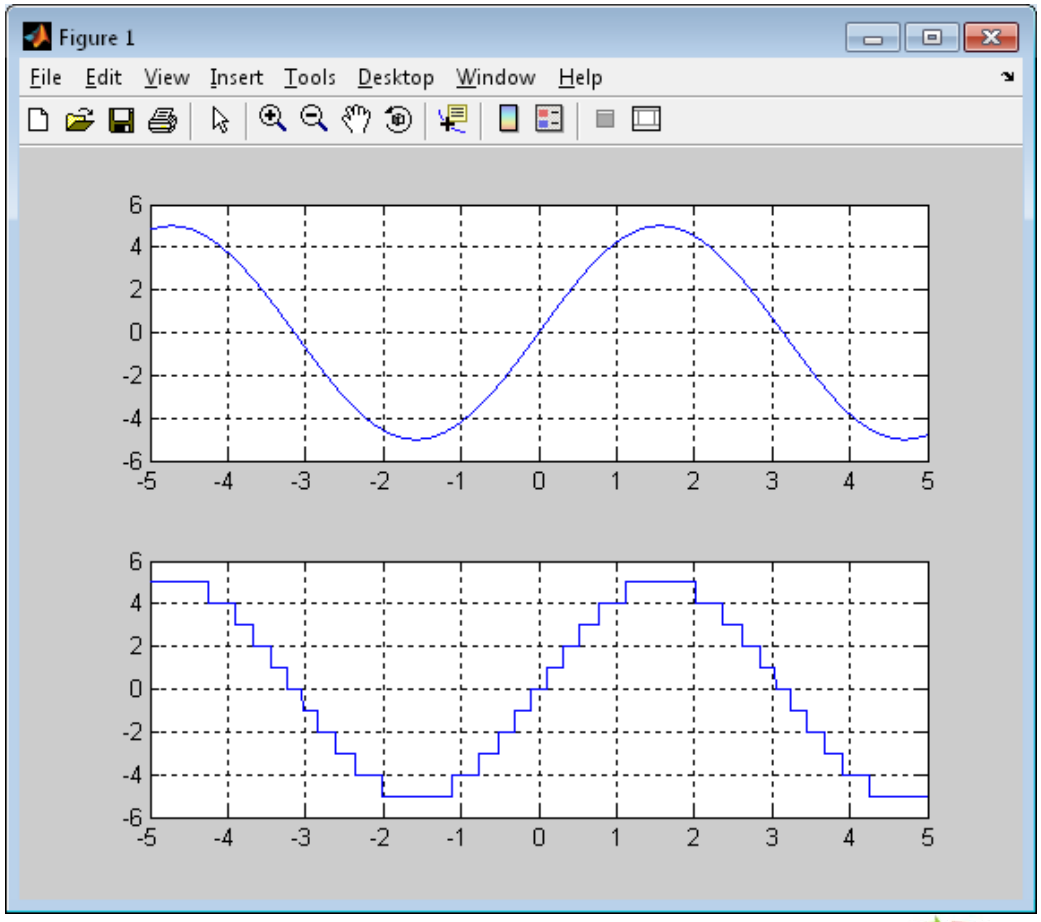
Sliding Window Filter – example



Computational errors

Floating Point Arithmetic vs Fixed Point Arithmetic

```
x = [-5:0.001:5];
y=sin(x)*5;
subplot(2,1,1);
plot(x,y);
grid;
ylim([-6, 6]);
subplot(2,1,2);
plot(x,int32(y));
grid;
ylim([-6, 6]);
```

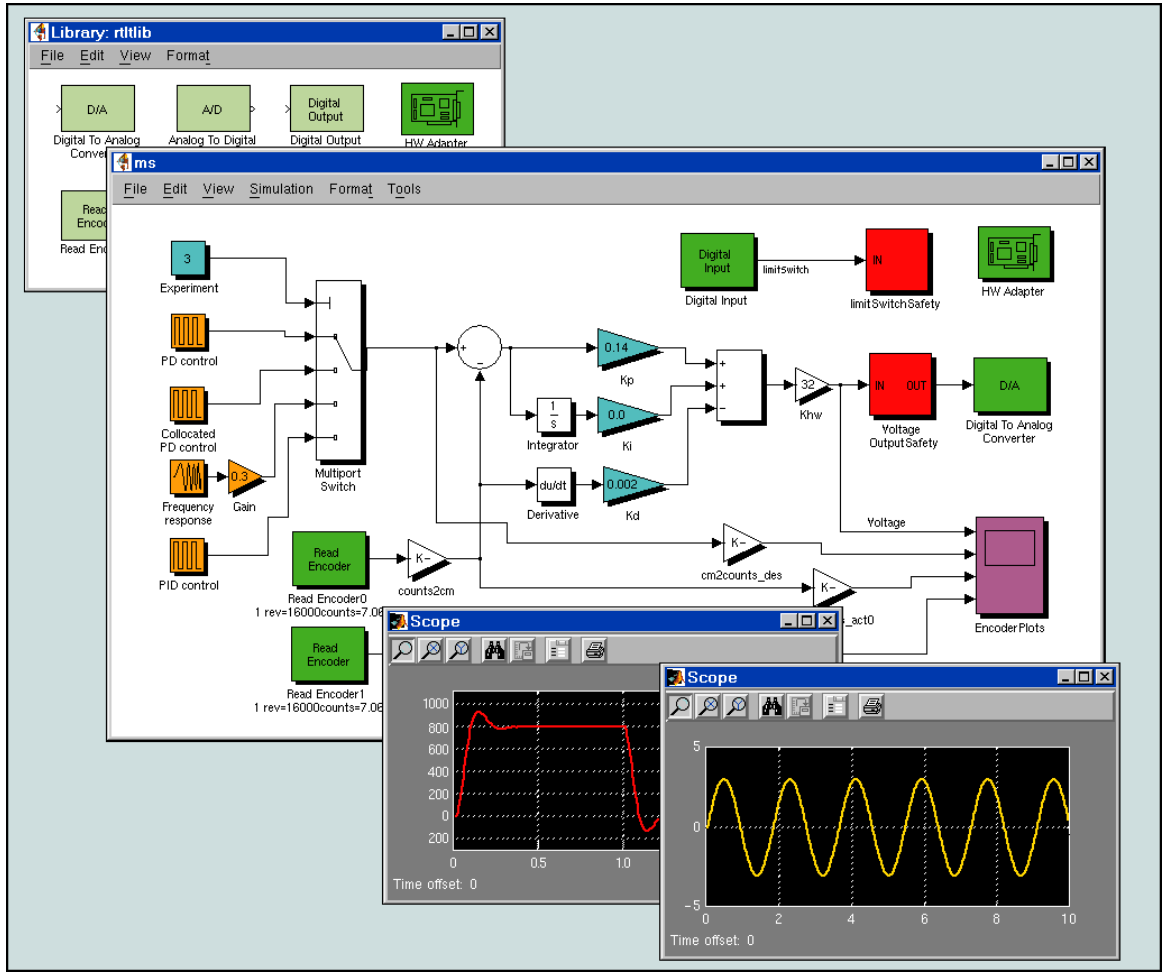


Simulink

MATLAB®
SIMULINK®

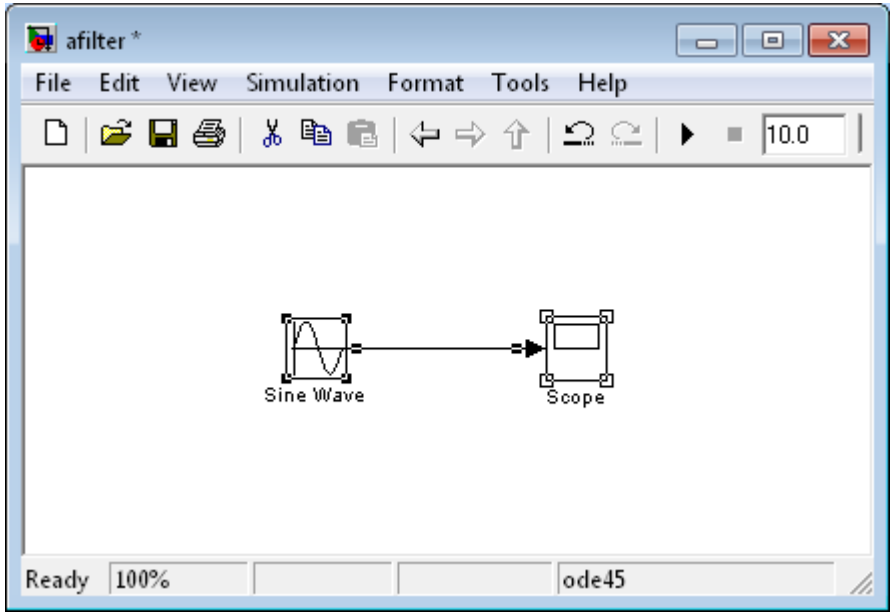
Current directory > new > model

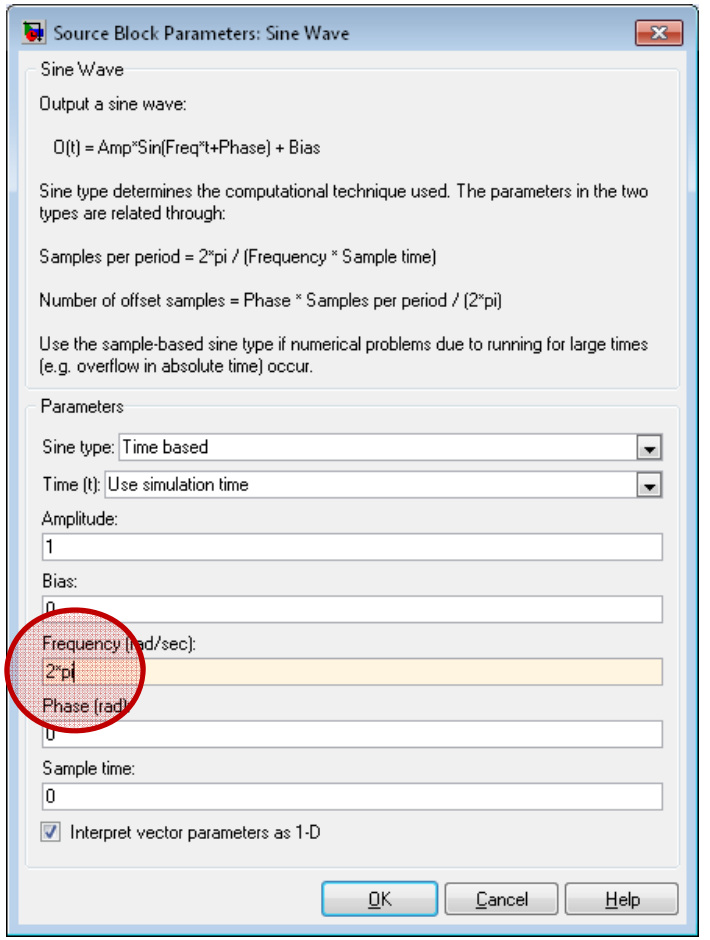
Simulink library



Simulink

Simulink > Sources > Sine wave
Simulink > Sinks > Scope

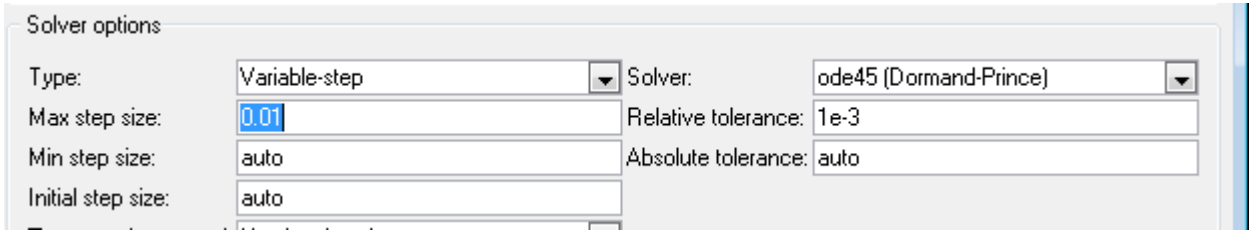




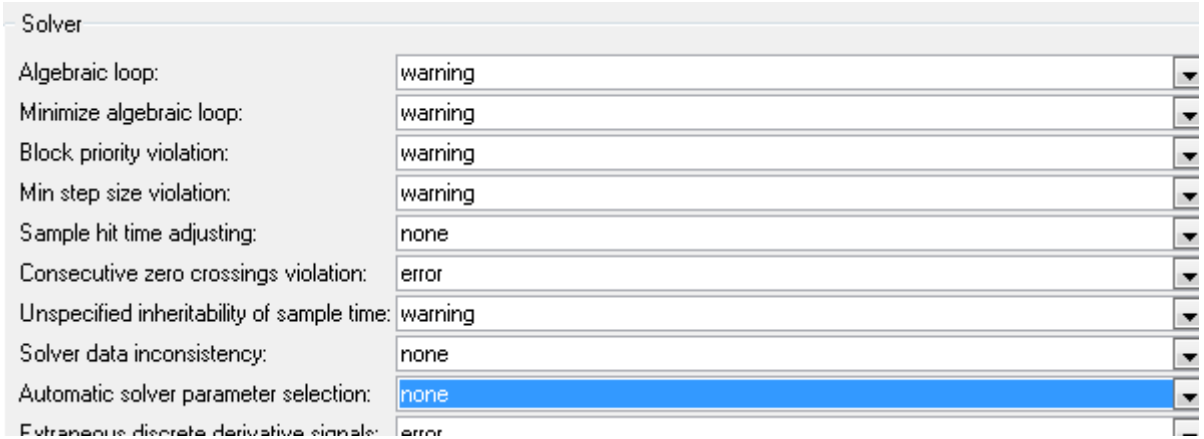
Simulink

Simulation > Configuration parameters >

- **Solver > Max step size: 0.01**

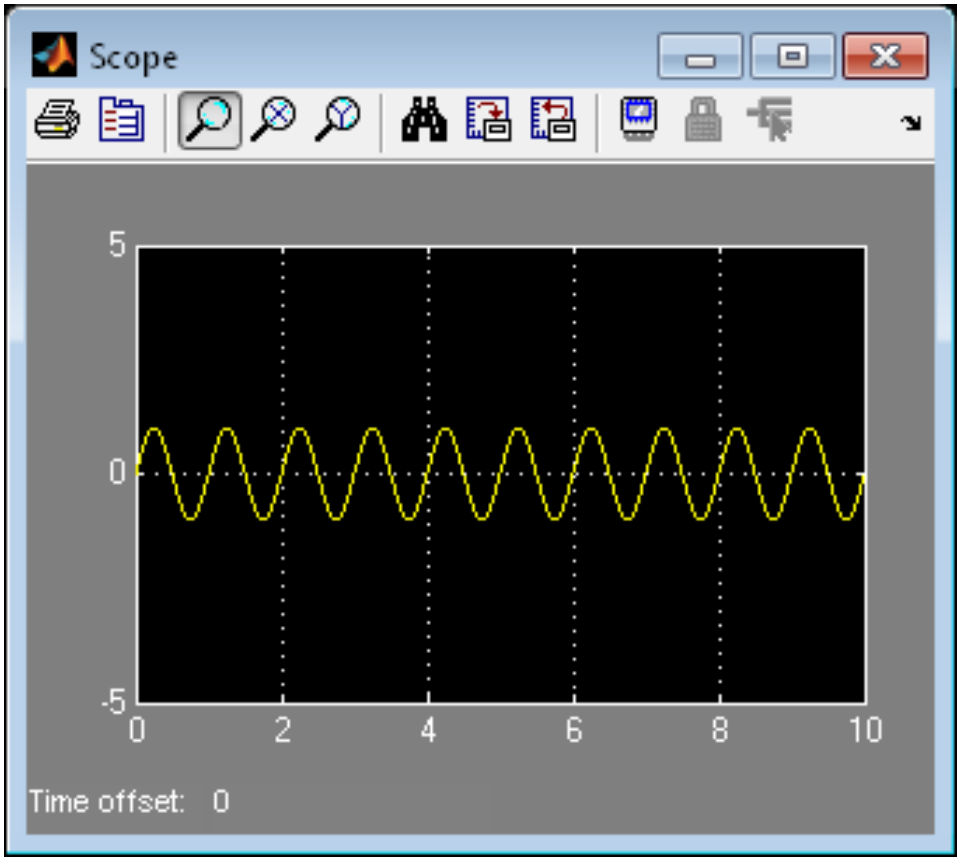


- **Diagnostics > Automatic solver parameter selection: none**



Simulink

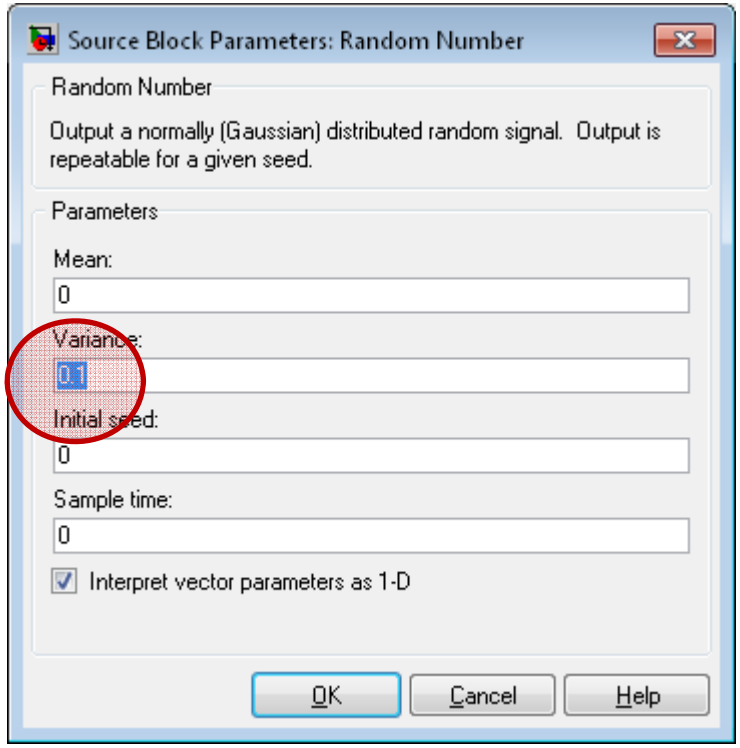
Run

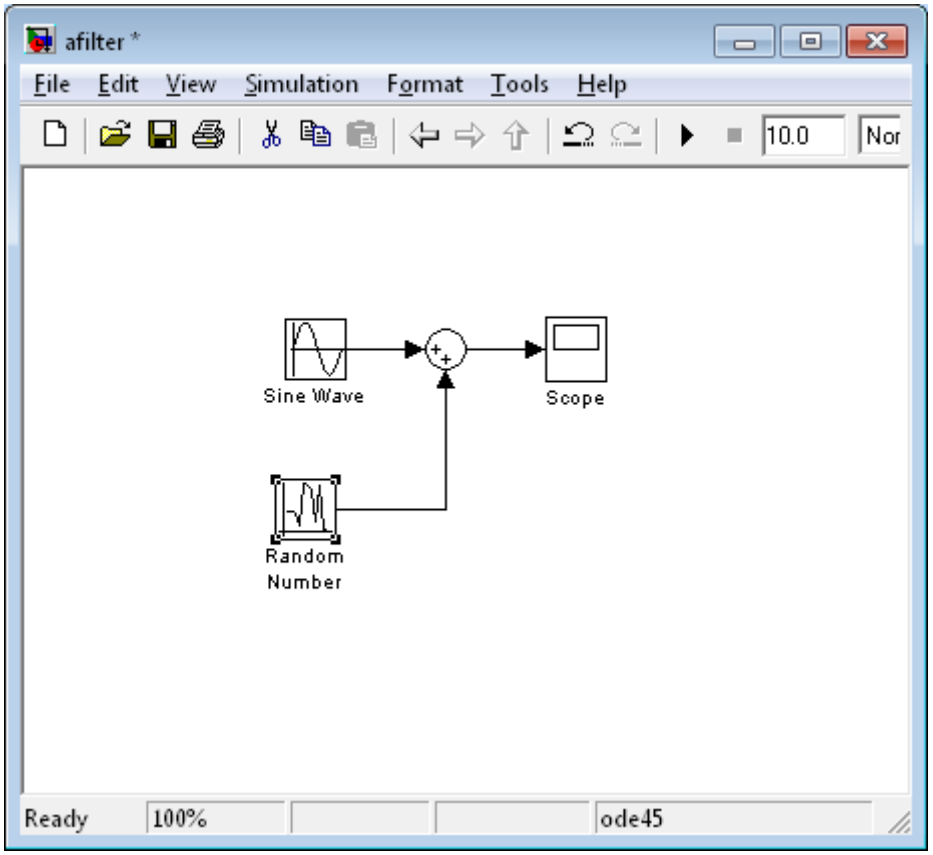


Simulink

Commonly used blocks > sum

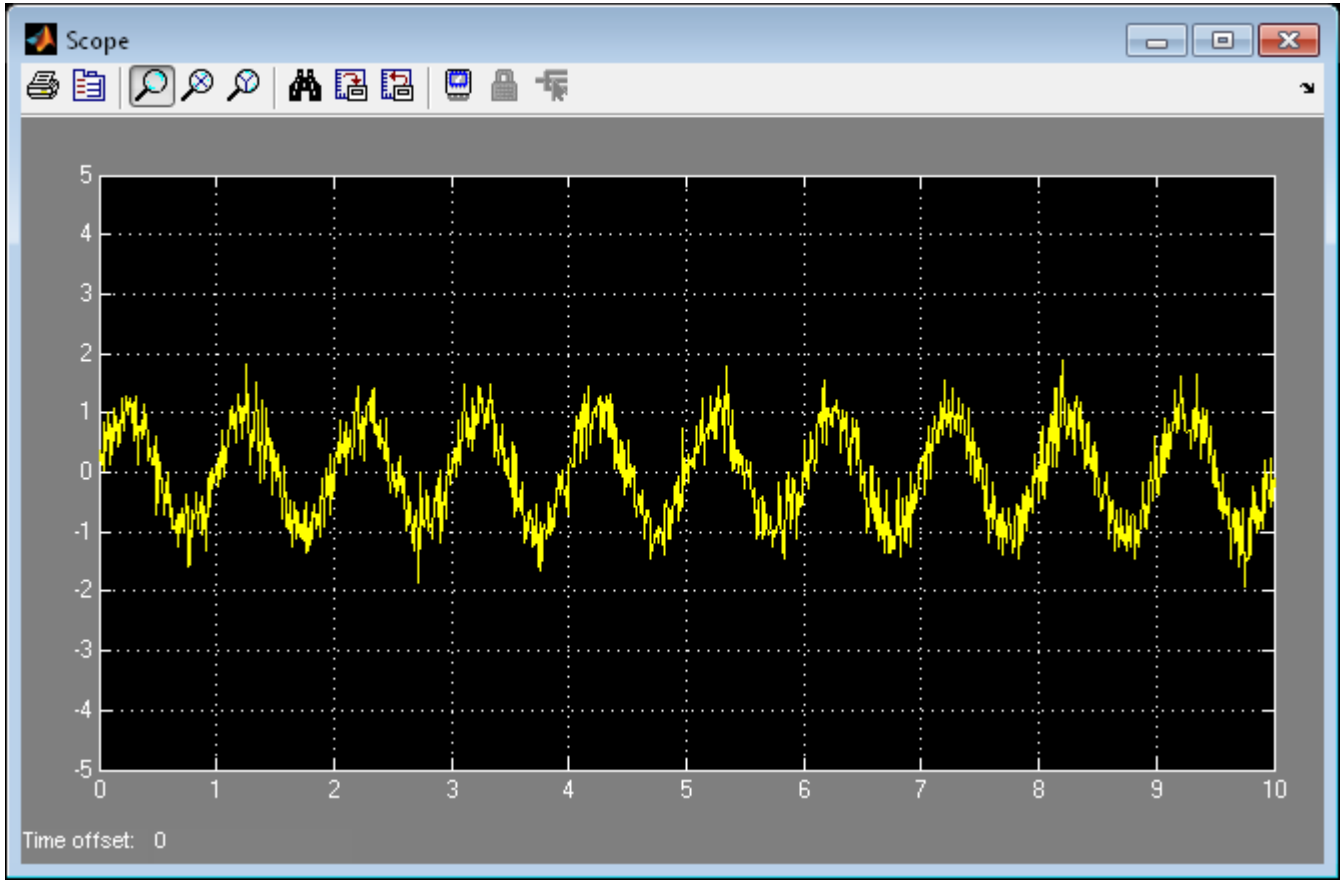
Sources > random number





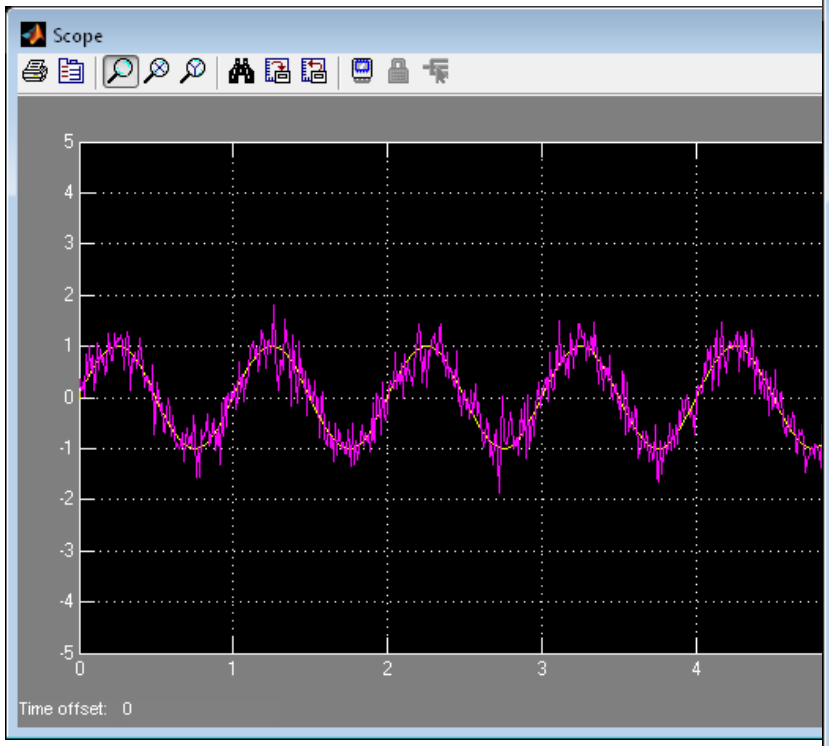
Simulink

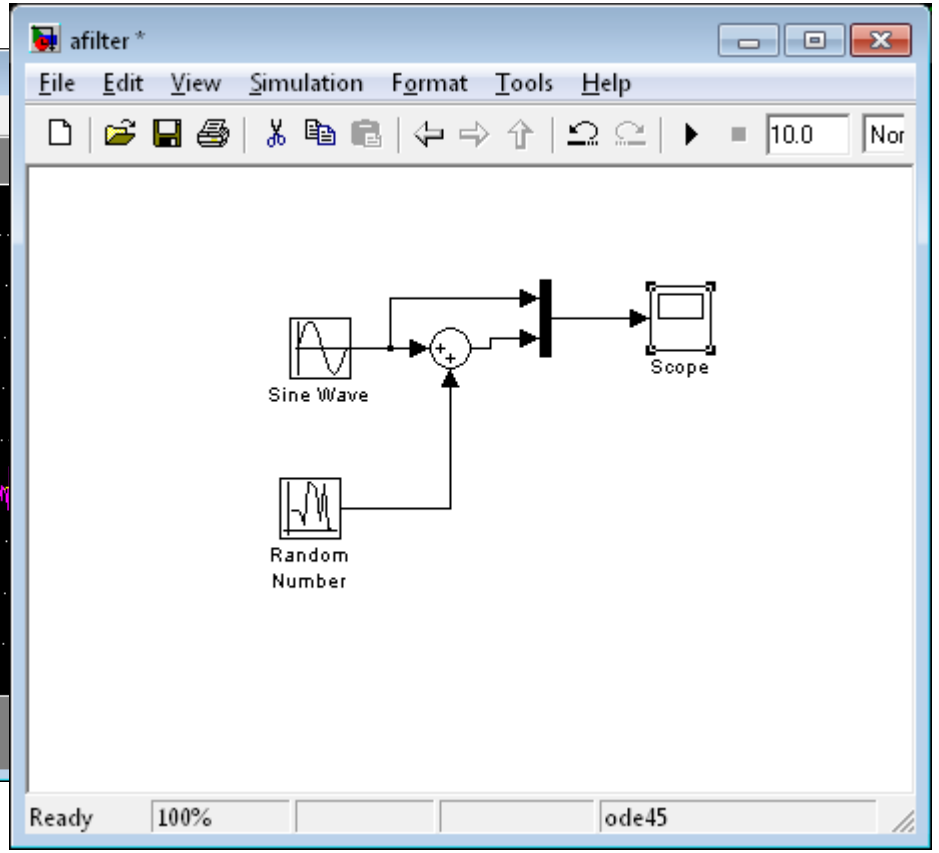
Run



Simulink

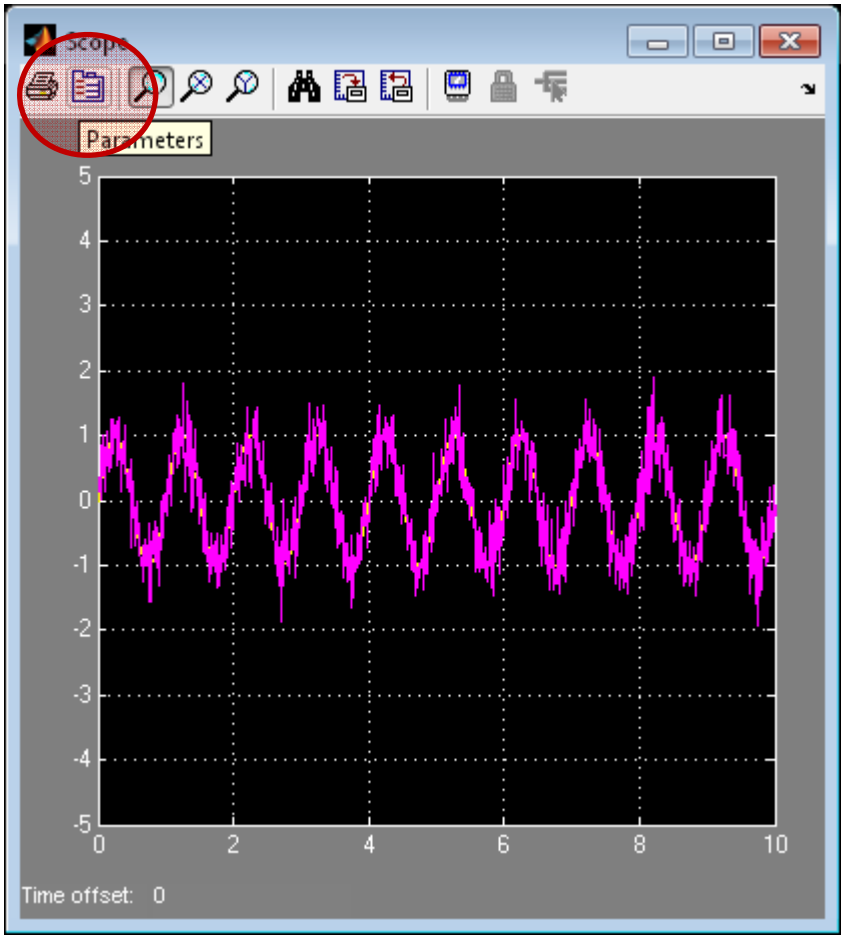
Commonly used blocks > Mux

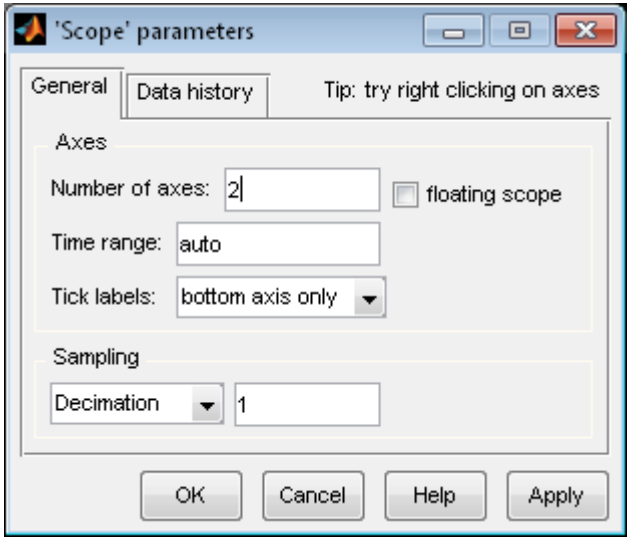




Simulink

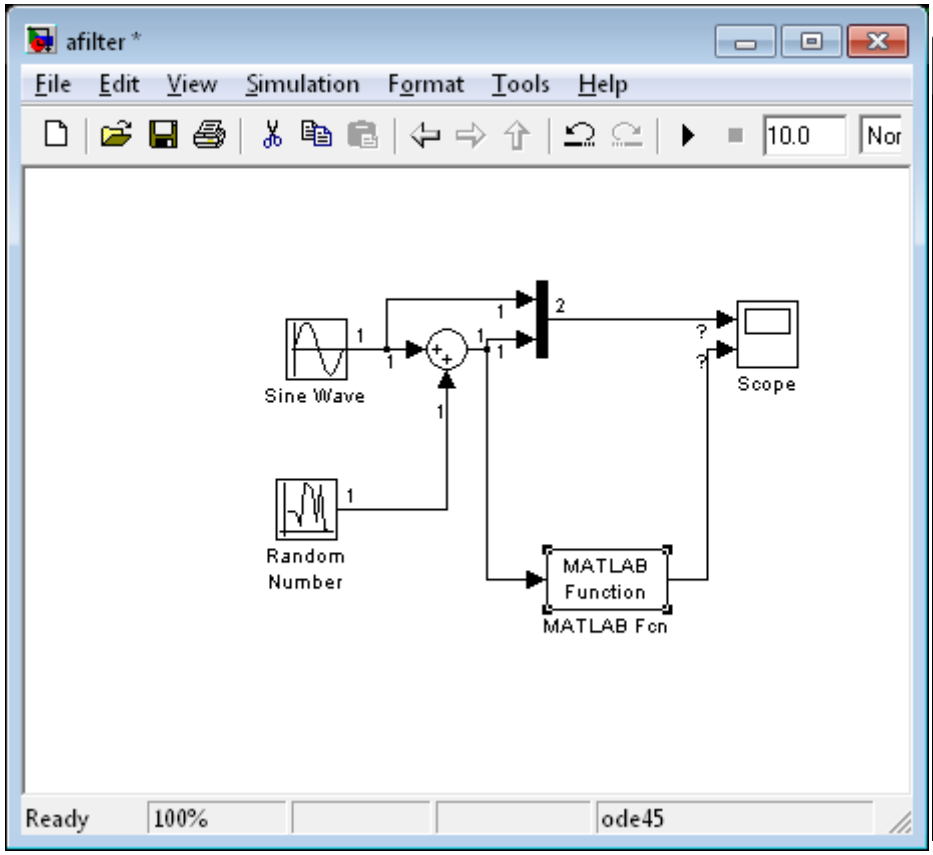
Parameters

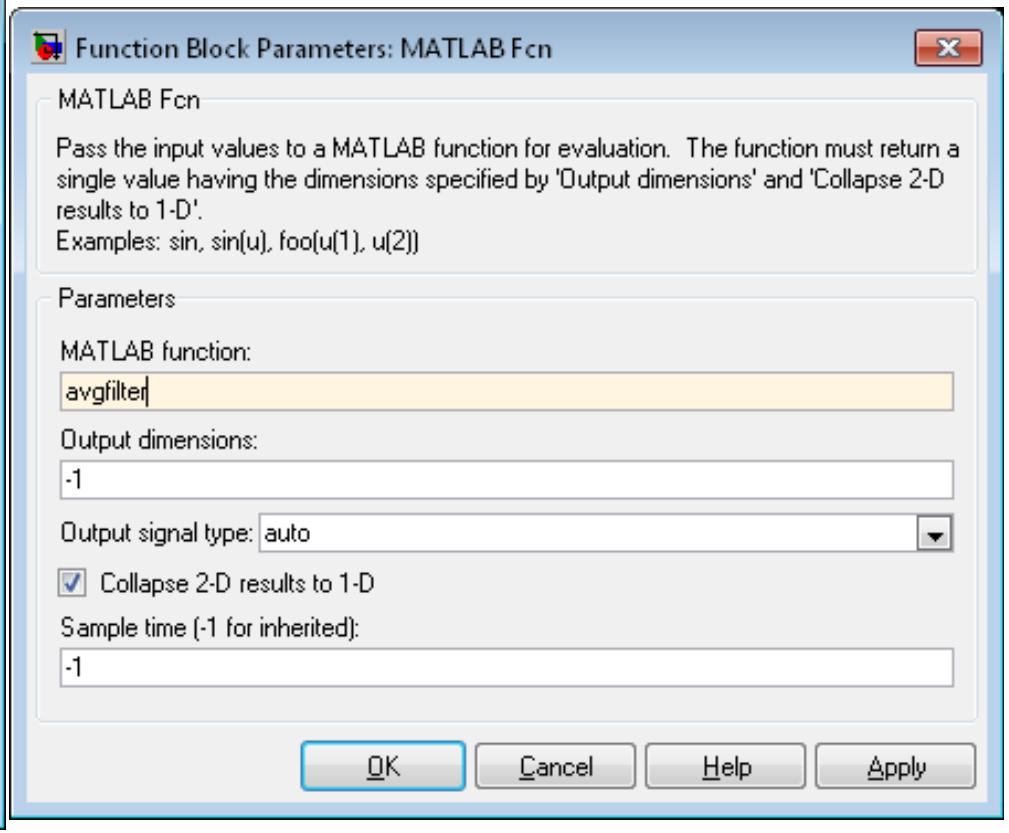




Simulink

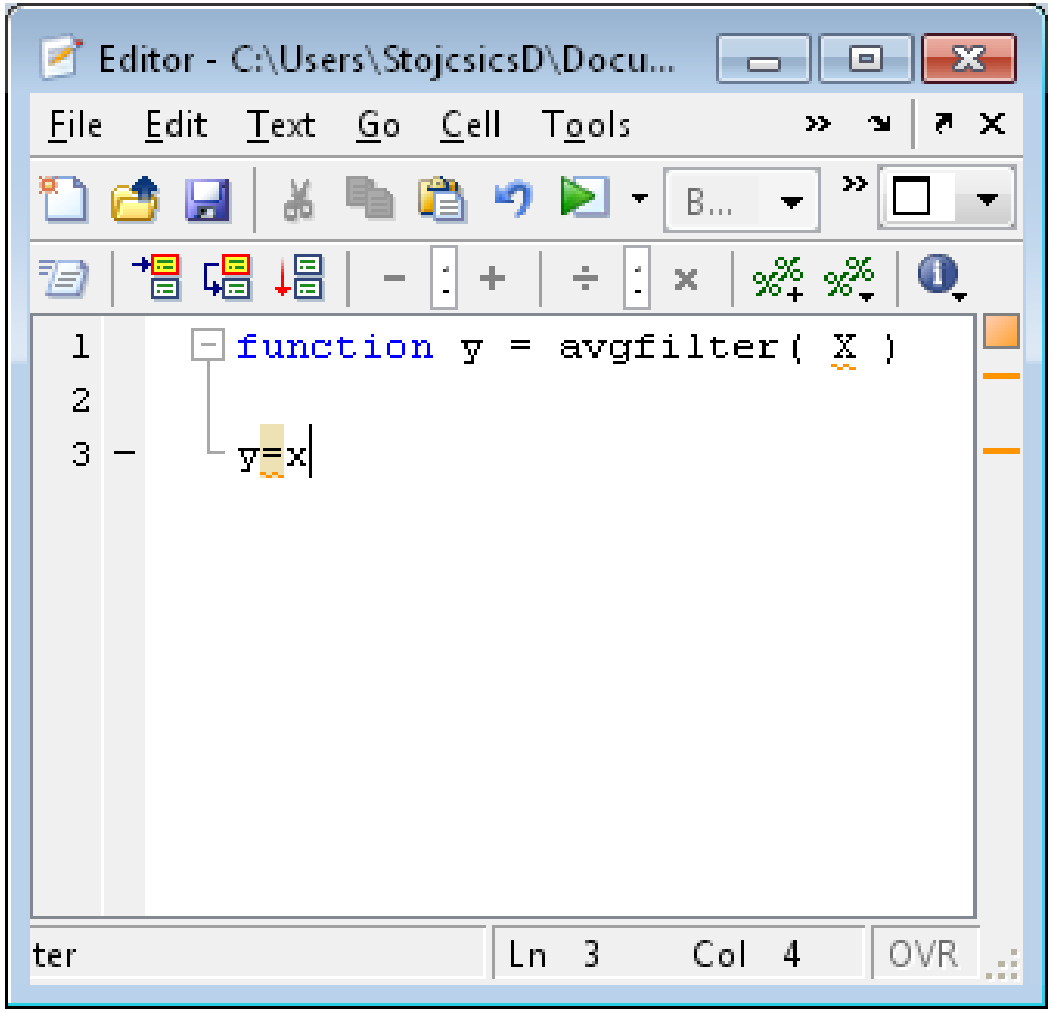
User-defined functions > MATALB fcn





Simulink

Create new m-file

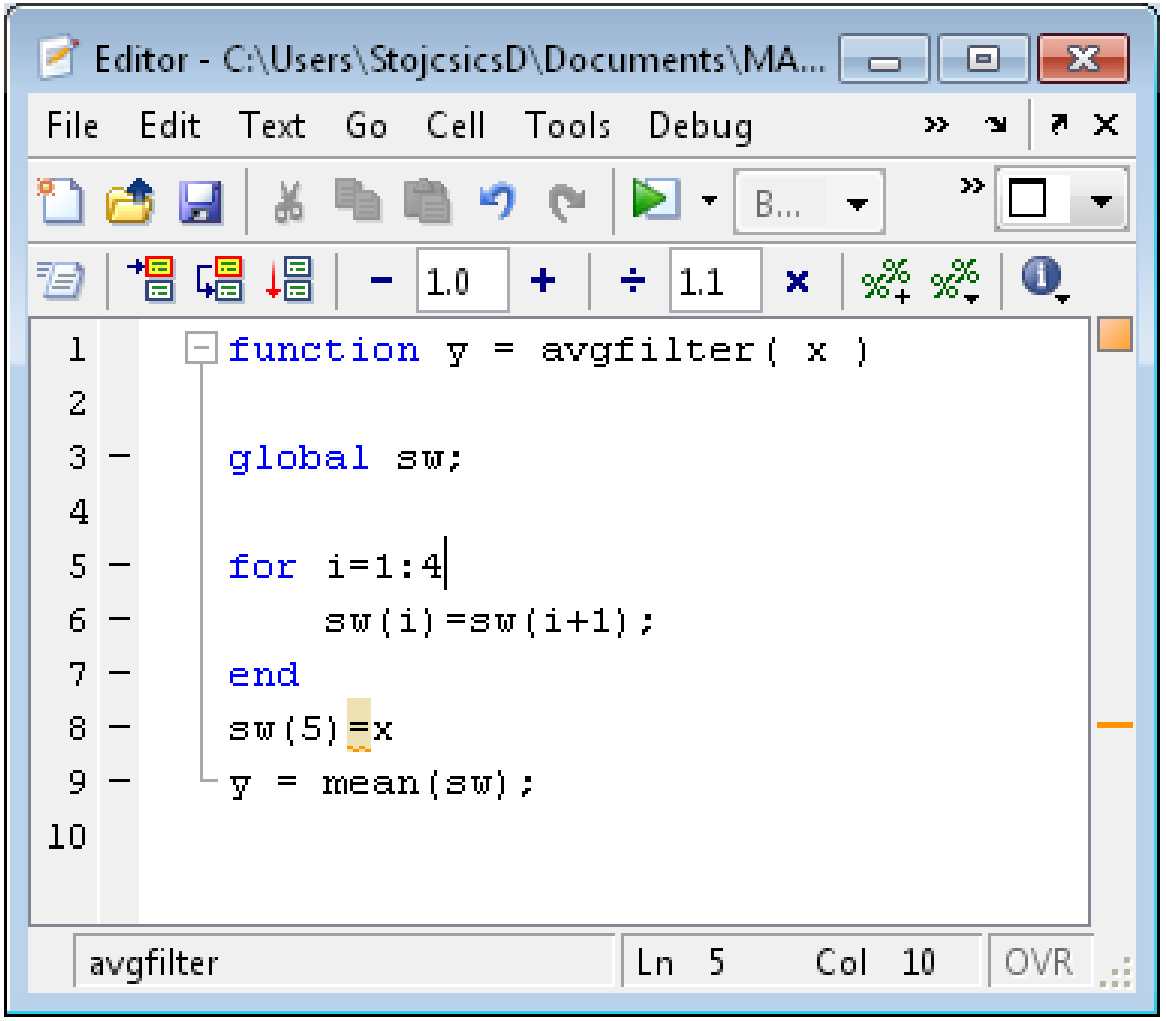


Simulink

Command window:

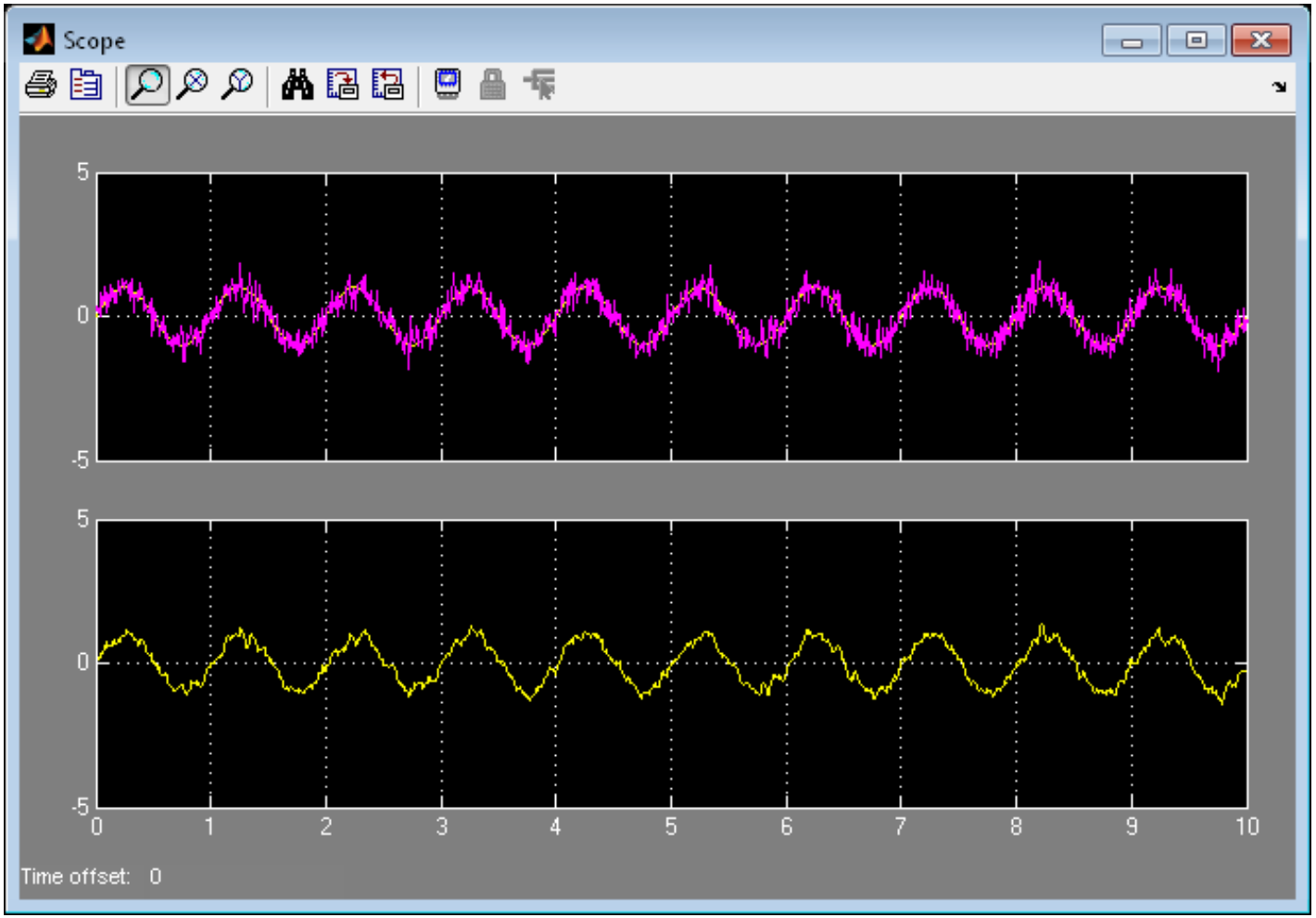
```
>> global sw
>> sw=[0,0,0,0,0]

sw =

     0     0     0     0     0
```

Simulink


Editor - C:\Users\StojcsicsD\Documents\MA...
File Edit Text Go Cell Tools Debug
1.0
1.1
1 function y = avgfilter(x)
2
3 global sw;
4
5 for i=1:4
6 sw(i)=sw(i+1);
7 end
8 sw(5)=x
9 y = mean(sw);
10
avgfilter
Ln 5 Col 10
OVR

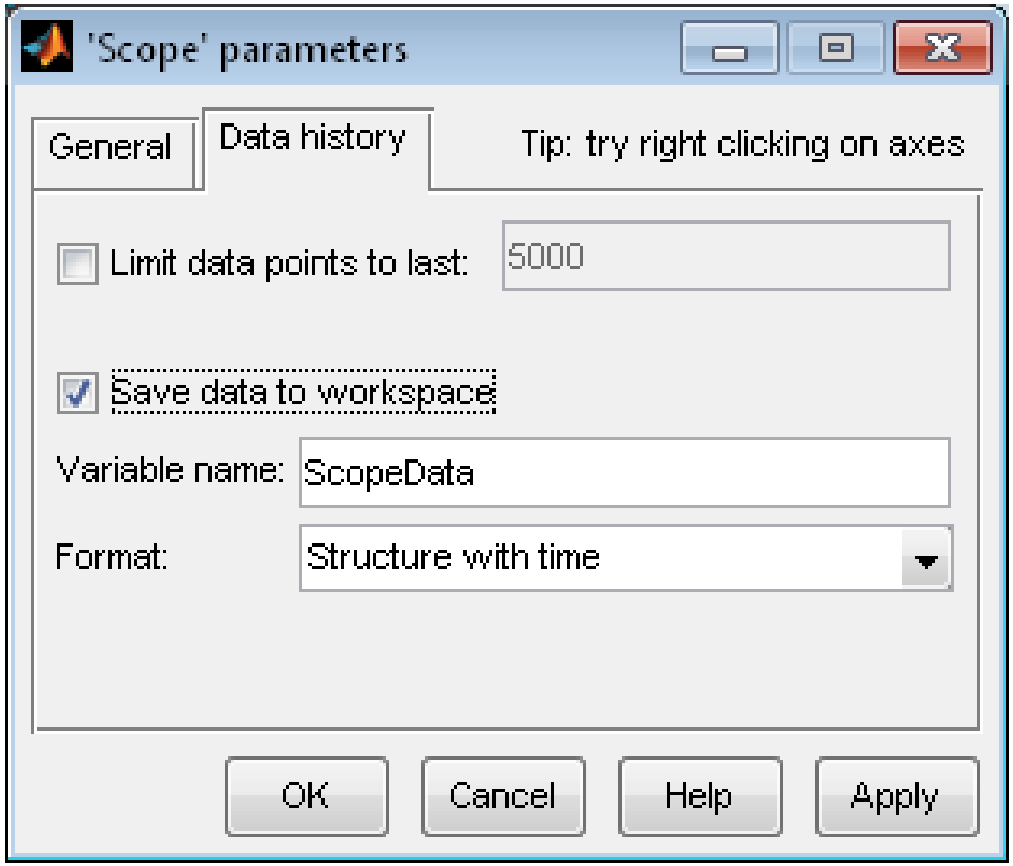
Simulink

Run



Simulink

Scope parameters

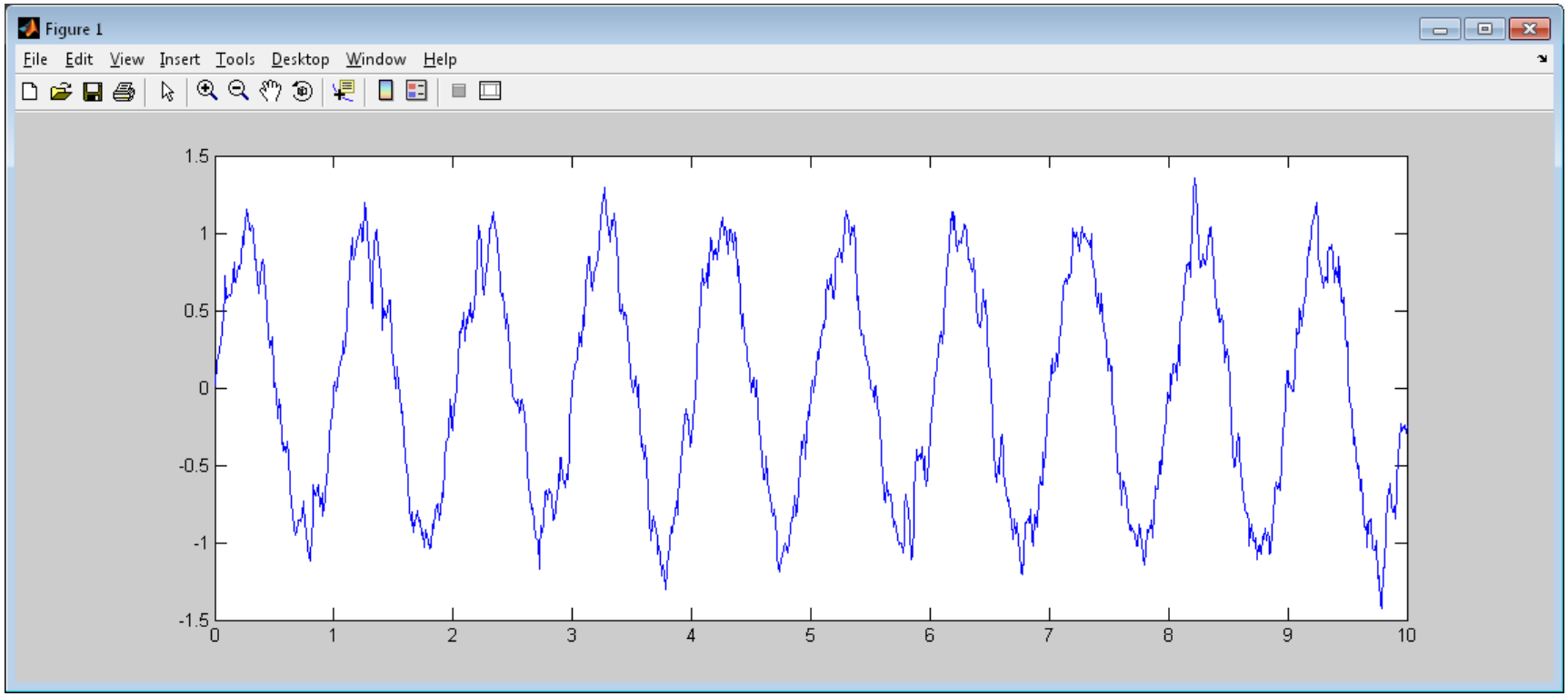


ScopeData.signals(1).values
ScopeData.signals(1).values(:,1)
ScopeData.signals(1).values(:,2)
ScopeData.signals(2).values
ScopeData.time

Simulink

Scope parameters

```
plot(ScopeData.time, ScopeData.signals(2).values)
```



Course unit title:
Basics of Information Systems
Course unit code:
NIRIA1SEND

Thank you for your attention!

kovacs.levente@nik.uni-obuda.hu

24.09.2013.