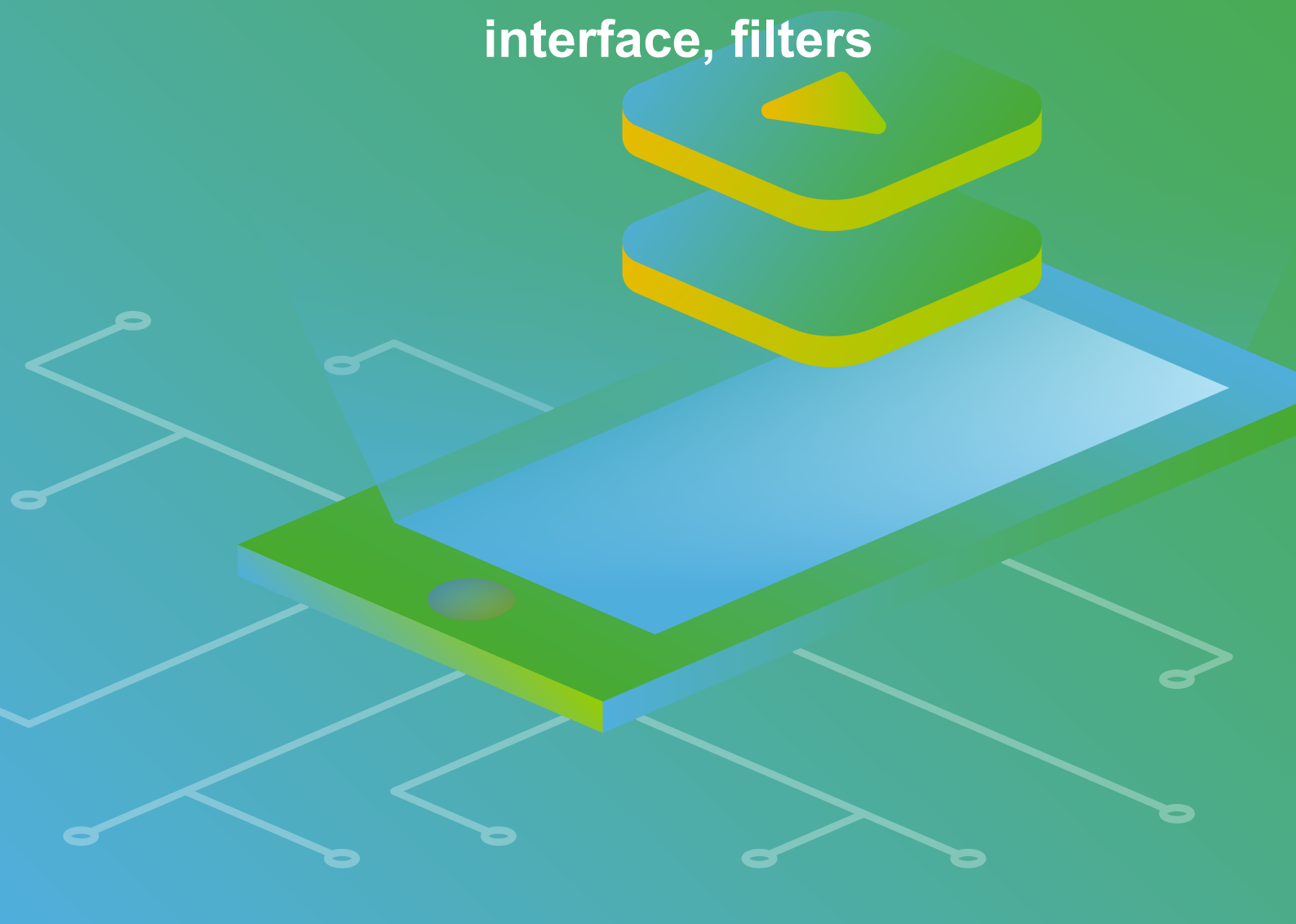
POLYTECH
Peter the Great
St.Petersburg Polytechnic
University

Course
« Introduction to Biomedical Engineering»

Dr. Kirill Aristovich

Section 4: Basics of High-level programming: Matlab
Lecture 4.2: Visualisation, serial interface, filters



Visualisation, serial interface, filters

The essential bit of any high-level programming, especially for science and prototyping, is the ability to represent the data graphically. Matlab offers a very simple and powerful tools for doing this. You can create a basic line graph in MATLAB using the function ‘plot’. ‘plot(y)’ will plot your data (y) against an arbitrary (numbered) x-axis, whereas ‘plot(x, y)’ will plot your data (y) against the corresponding x-axis locations in x. If y is a matrix, the ‘plot’ function will display all columns of y as separate lines, and if your data is arranged in rows, use the apostrophe shortcut to transpose the matrix.

- You can create a **basic line graph** in MATLAB using the function ‘**plot**’
- **‘plot(y)’** will plot your data (**y**) against an arbitrary (**numbered**) x-axis
- **‘plot(x, y)’** will plot your data (**y**) against the corresponding x-axis locations in **x**
- If y is a **matrix**, the ‘**plot**’ function will display all **columns** of y as **separate lines**
- For data is **arranged in rows, use the apostrophe** shortcut to **transpose the matrix**
- You can create a **basic line graph** with **error bars** using the function ‘**errorbar**’
- ‘**errorbar(x, y, SE)**’ will plot data with **error bars** whose **length is specified by SE**
- **Scatter plots** can be created in MATLAB using the ‘**scatter**’ function
- The ‘**scatter**’ function requires a minimum of **two inputs** (i.e. both **x** and **y** data)

Figure 1 - Basic Plotting

You can create a basic line graph with error bars using the function ‘errorbar’. Also, scatter plots can be created in MATLAB using the ‘scatter’ function.

The ‘plot’ function automatically generates a new figure window, if one isn’t open yet. However, it automatically overwrites an existing figure window. To open an additional figure window, use the ‘figure’ command. To add lines to an existing figure, you can use the ‘hold on’ / ‘hold off’ commands. You can close figures individually with the ‘close’ command, or close them all by executing ‘close all’.

MATLAB automatically scales figure axes to accommodate the data being plotted, but you can control this manually using ‘xlim’ and ‘ylim’ commands. There are several functions for adding labels, titles and text to figures. Text added by these functions can also be formatted directly from the command line. This is achieved by passing additional inputs to the function. There are two ways to edit the properties of a figure. The first is through the figure window toolbar, by clicking Edit -> Figure Properties. The second is directly from the command line.

- MATLAB **automatically scales** figure axes to **accommodate the data** being plotted
- Axes limits can be **manually controlled** using '**xlim([min max])**' and '**ylim([min max])**'
- The '**axis**' function can also be used adjust axes in several different ways
- '**axis tight**' fits the axes limits to the data exactly
- '**axis square**' makes the figure axes square, for display purposes
- '**axis off**' removes the axes, labels and background, for display purposes
- Each of these axis limit commands can be used on **any MATLAB plot**

Figure 2 - Axis Limits

Basic properties can be defined when first creating the plot, with the specific short-codes for color, marker size, and line styles. More complex properties can be adjusted using the 'set' function. The 'figure handle' here identifies the figure that we wish to edit, and you can use the shortcut 'gca', which specifies the 'current' figure.

Several different panels can be plotted within a single figure using 'subplot'. Panels are numbered along each row, for example, subplot(2,2,1) sets up a 2 by 2 square of panels, and plots in the first. Plots can also be spread across multiple panels. When drawing subplots, figure handles can be used to control each panel separately, assigning variable to each panel. This is achieved by providing the 'subplot' function with an output variable, like so. Properties of that specific panel can then be controlled using command 'set'.

Vertical and horizontal bar charts can be created using the 'bar' and 'barh' functions, and histograms can be created using the 'hist' function. Two dimensional plots can be created using 'contourf' and 'imagesc'. Note that the input for those is the matrix where each value represents a color to plot. There are many other different plotting functions within MATLAB. Each of these can be controlled using the general syntax discussed above.

MATLAB 'Color' Code

'b'	= Blue (default)
'g'	= Green
'r'	= Red
'k'	= Black
'y'	= Yellow
'w'	= White
'c'	= Cyan
'm'	= Magenta

MATLAB 'LineStyle' Code

'-'	= Solid line (default for plot)
'--'	= Dashed line
'-.'	= Dash-dot line
':'	= Dotted line
'none'	= No line

MATLAB 'Marker' Code

'o'	= Circle (default for scatter)
'.'	= Point
'x'	= Cross
'+'	= Plus sign
'*'	= Asterisk
's'	= Square
'd'	= Diamond
'v'	= Down triangle
'^'	= Up triangle
'>'	= Right triangle
'<'	= Left triangle
'p'	= Pentagram
'h'	= Hexagram
'none'	= No marker (default for plot)

Figure 3 - Editing Figure Properties

Moving on, it is useful to discuss programming practices. The basics are: Assign variables early, make your code easily extendable and adaptable. Go from simple to

complex: If you are faced with a complex issue, develop a toy model first. Use modular programming: create functions when possible. Back-up your code before you have change it. Make extensive use of comments! And finally, don't have faith! Always test your code with simple examples.

It is also important to optimize your code in terms of memory and processor requirements. Some basic rules can help you to avoid sluggish script and may play a decisive role in making you script runnable at all: Pre-assign arrays and don't let them grow inside of loops. Avoid loops when possible (they are slow), favor matrix algebra. Favor logical operators over the built-in functions "find" and "nonzeros", and consider using sparse matrices (although note that sparse indexing is also quite slow).

- Optimizing your code is important for large projects. A very wide topic.
- But even for smaller projects a few basic guidelines should be followed.
- Pre-assign arrays and don't let them grow inside of loops
- Avoid loops when possible (are slow), favor matrix algebra
- Favor logical operators over the built-in functions "find" and "nonzeros"
- Consider using sparse matrices (but sparse indexing is slow)

Figure 4 - Memory and Performance

In the end, let's talk about some specific matlab features which are essential for Embedded system –to - computer interface. We ARE going to be using them in the final section.

The convenient way for communicating to your microcontroller, as we discussed previously, is the serial interface. Here is the set of commands that will enable this functionality from the computer side. Essentially, you need to open the serial port, set up the baud rate, and open the serial. Make sure that bode rate matches the one on the controller side!

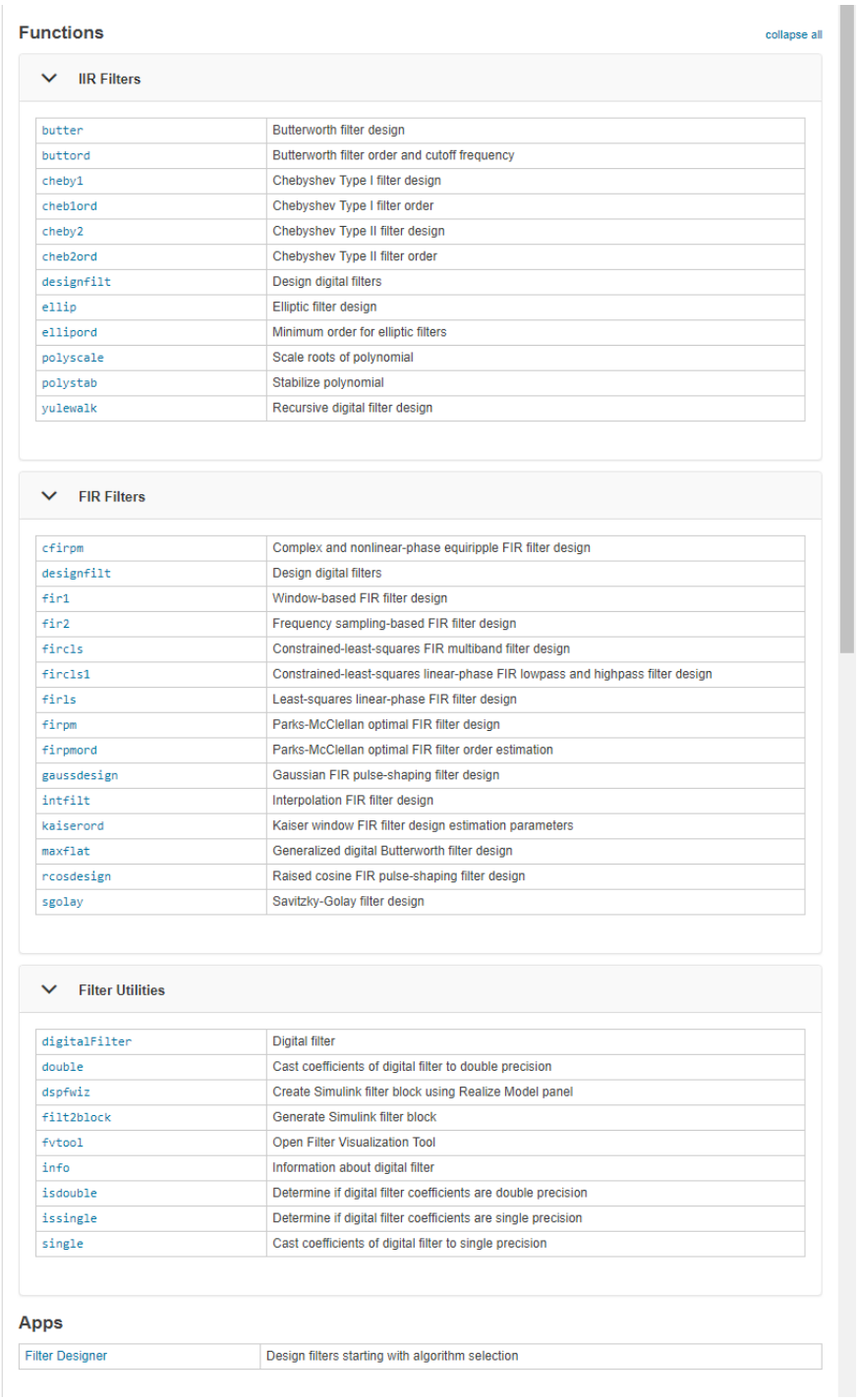
```
s = serial('COM1');          % Creates serial object
set(s,'BaudRate',4800);      % sets Baud Rate to 4800
fopen(s);                    % opens serial port
fprintf(s,'*IDN?')           % sends 'who are you?' command
out = fscanf(s);             % reads from buffer into variable  out
fclose(s)                    % closes the port
delete(s)                    % deletes the object
clear s                      % clears the variable
```

Figure 5 - Matlab serial interface

Then you can transmit the stuff, or read a value from a buffer into a particular variable. Don't forget to close the serial and clear it after you've done with it – otherwise it will stay open and you would not be able to open it again if matlab is still running.

The useful commands to execute closing everything is ‘close all’ and ‘close(InstrFind)’ if you want to shut down everything forcefully.

Finally, it is worth noting the digital filtering toolbox, which offers a powerful set of tools to get your signal filtered. This does not replace antialiasing filter! However for everything else you can use it to essentially emulate the analog filters, and sometimes even over-perform them. The stuff comes at a cost of processing power, but since in most cases it is not a concern for modern computers, all of this tools can operate real-time these days.



Functions collapse all

IIR Filters

butter	Butterworth filter design
buttord	Butterworth filter order and cutoff frequency
cheby1	Chebyshev Type I filter design
cheb1ord	Chebyshev Type I filter order
cheby2	Chebyshev Type II filter design
cheb2ord	Chebyshev Type II filter order
designfilt	Design digital filters
ellip	Elliptic filter design
ellipord	Minimum order for elliptic filters
polyscale	Scale roots of polynomial
polystab	Stabilize polynomial
yulewalk	Recursive digital filter design

FIR Filters

cfirpm	Complex and nonlinear-phase equiripple FIR filter design
designfilt	Design digital filters
fir1	Window-based FIR filter design
fir2	Frequency sampling-based FIR filter design
fircls	Constrained-least-squares FIR multiband filter design
fircls1	Constrained-least-squares linear-phase FIR lowpass and highpass filter design
firls	Least-squares linear-phase FIR filter design
firpm	Parks-McClellan optimal FIR filter design
firpmord	Parks-McClellan optimal FIR filter order estimation
gaussdesign	Gaussian FIR pulse-shaping filter design
intfilt	Interpolation FIR filter design
kaiserord	Kaiser window FIR filter design estimation parameters
maxflat	Generalized digital Butterworth filter design
rcosdesign	Raised cosine FIR pulse-shaping filter design
sgolay	Savitzky-Golay filter design

Filter Utilities

digitalFilter	Digital filter
double	Cast coefficients of digital filter to double precision
dspfwiz	Create Simulink filter block using Realize Model panel
filt2block	Generate Simulink filter block
fvtool	Open Filter Visualization Tool
info	Information about digital filter
isdouble	Determine if digital filter coefficients are double precision
issingle	Determine if digital filter coefficients are single precision
single	Cast coefficients of digital filter to single precision

Apps

Filter Designer	Design filters starting with algorithm selection

Figure 6 - Filters. Filtered = filtfilt(F, Unfiltered);

The syntax is also quite simple. You first need to create a specific filter. Depending on the application there can be many varieties. You can use simple butterworth filter of n-th order, giving it the order and cut-off frequencies, or use the sophisticated ‘designfilt’ command. This even have a user interface and lets you choose among different filter types and configurations, while at the same time control cut-off frequencies and ripples.

You need to output the filter to a variable, and then use this variable with the ‘filtfilt’ command. You give filtfilt the filter variable, and the data which needs to be filtered, and it outputs already filtered data.

If you use the matrix data, then columns of the matrix will be filtered independently. It essentially treats the first variable as time, and second as independent channel number.

All right, now we have learned everything that we need, I will see you on the final lecture where we will finally design and implement the active bionic prosthetics!

This course was developed with the support of
the "Open Polytech" educational project

Online courses from the top instructors of SPbPU