MATLAB-Programming for Engineers: Objects and Stream processing Lecture 5

Dr. Zsolt Kollár

BME HVT
http://hvt.bme.hu

2020

Table of contents

Generally about objects

Objects have

- properties
- methods
- callback functions – event driven execution

These properties/methods can be public or private. If it can be seen and accessed from outside the object.
In MATLAB hardware interfaces, timers and graphics are handles trough object. The properties can be accessed or modified using `get` and `set` functions or as structure variables.

Callback functions

A callback function can be

- A function handle
- A cell array containing a function handle and additional arguments if needed
- A character vector that evaluates to a valid MATLAB expression.

A callback function is generally defined with 2 default inputs: The `Object` corresponding to the current object for which the callback function was called and an `Event` structure which contains some specific data.

```
function myCallback(Object, Event)
% Object — Handle to current object
% Event — Specific information related the object

% Function code

end
```

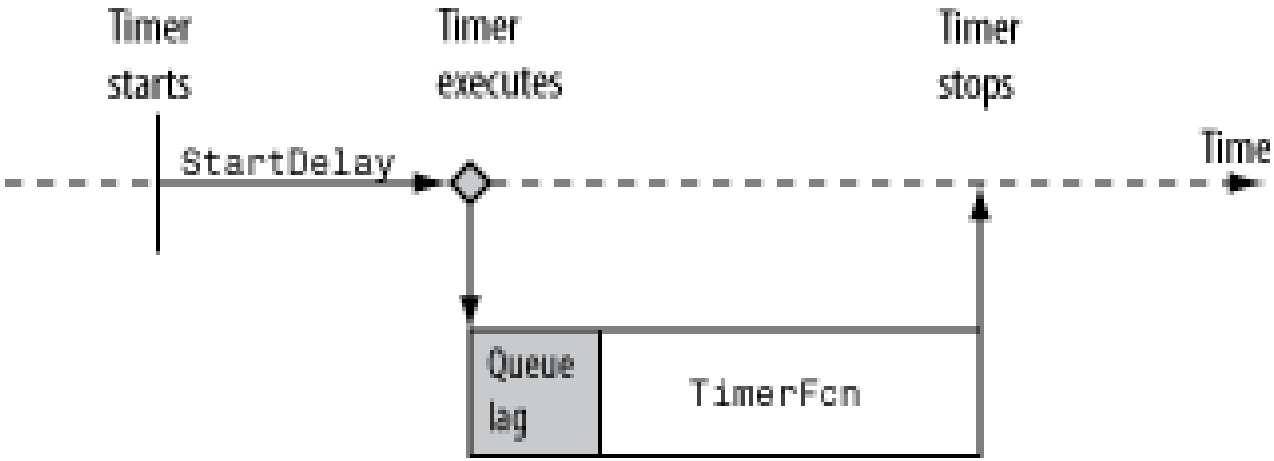
Audio object versus function

Try `audiotest.m`! Demonstration on how to use the `audioplayer` object.

```
load handel

%% Using fuction
sound(y,Fs);                    % Play music
pause(length(y)/Fs);            % Wait until music stops

%% Using object
audioH = audioplayer(y,Fs); % Create object
get(audioH);                    % See all properties of the object

audioH.play;                    % Start playing
pause(5);                       % Wait 5 secondes

% pause - double rate - resume
audioH.pause;
audioH.SampleRate = 2*audioH.SampleRate;
audioH.resume;
```

Timer object is used to measure, schedual and execute a MATLAB code in a given timer period automatically using callback functions.

singleShot



Try `timertest.m`! Set up timer and use callback with different definition examples

```
%% Setup timer
t = timer;                % Create new timer
t.StartDelay     = 5;     % Delay timer start with 5 seconds
t.Period         = 2;     % Set timer for 2 seconds
t.TasksToExecute = 3;     % Execute timer 3 times
t.ExecutionMode  = 'fixedRate'; % Execute using fixed rate

%% Callback functions definition examples
% Function handle for Timer start
t.StartFcn = @timercallback;
% Anonym function handle for Timing instant
t.TimerFcn = @(~,thisEvent) disp([thisEvent.Type ' executed '...
     datestr(thisEvent.Data.time,'dd-mmm-yyyy HH:MM:SS.FFF')]);
% Character string for Timer stop
t.StopFcn = 'disp([''StopFcn executed '', datestr(clock,''dd-mmm-yyyy HH:MM
     :SS.FFF'')])';

%% Start timer
start(t)
```

Graphical objects: Setting properties

Try `graphicsstest1.m`!

- Create a new figure
- Set some properties of the figure through the figure handle

```
close all;
%% Create new figure
FigH = figure;  % Save figure handle
set(FigH);      % See all properties with possible values

% Set some properties
set(FigH,'Color','y');              % Set color using set function
FigH.Name       = 'First figure';   % As a structure
FigH.ToolBar    = 'none';           % As a structure
FigH.MenuBar    = 'none';           % As a structure
FigH.NumberTitle='off';             % As a structure
```

Graphical objects II: Use callbacks

Try `graphicsstest2.m`!

- Create new figure.
- Set up some callback functions
- Try the callback functions

```
close all;
%% Create new figure
FigH = figure;  % Save figure handle
% Setup some callback functions
% Callback function if figure is closed -> Question dialog will apear
FigH.CloseRequestFcn   = @figurecallback;
% Callback function if keyboard key is pressed -> Name changes
FigH.WindowKeyPressFcn = @(src,event) set(src,'Name',src.CurrentCharacter);

% Plot a line and add a callback function for the line if clicked
plot(rand(1,5),'ButtonDownFcn',@lineCallback);
function lineCallback(obj,~)
   obj.Color = rand(1,3);
end
```

TCP/IP object

- Launch 2 MATLAB simultanously!
- Start `startserver.m` in one of them!
- Start `startclient` in the other one! It automatically send data to the server!
- Read data out of the server and convert it to characters using: `char(fread(server,server.bytesavailable))`

```
% Configure server object
server = tcpip('localhost', 30000, 'NetworkRole', 'server');
disp('Waiting for client...')
fopen(server)
```

```
% Configure client object
client = tcpip('localhost', 30000, 'NetworkRole', 'client');
fopen(client); % open client

pause(5); disp('Sending data');
fwrite(client, ['Hello server', char(10)]);

fclose(client); % close client
```

TCP/IP object with callback

- Launch 2 MATLAB simultanously!
- Start `startserver_callback.m` in one of them!
- Start `startclient` in the other one! It automatically send data to the server!

```
% Setup server
server = tcpip('localhost', 30000, 'NetworkRole', 'server');
set(server, 'BytesAvailableFcn',@server_callback) % Set callback function
fopen(server) % start server
% waiting for connection and for callback funtion to execute...

%% Callback function
function server_callback(obj, event)  % object, time
% Which event triggered the callback
event
event.Data
char(fread(obj,obj.BytesAvailable))
disp('The callback function was called!')
end
```

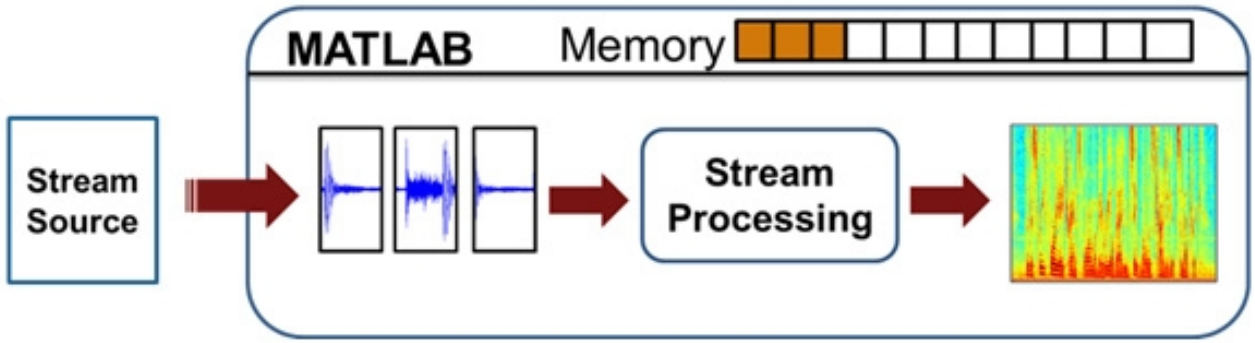
- Serial port communication can be handled in a similar way as TCP/IP

```
% To construct a serial port object:
s1 = serial('COM1');
s2 = serial('COM2', 'BaudRate', 1200);

% To connect the serial port object to the serial port:
fopen(s1)
fopen(s2)

% To query the device.
fprintf(s1, '*IDN?');
idn = fscanf(s1);

% To disconnect the serial port object from the serial port.
fclose(s1);
fclose(s2);
```

Table of contents

Basic idea of Stream processing

- Basic idea: quasi real-time signal processing
- Divide data into frames
- Process each frame before the next one arrives
- Applications: audio and wireless signal processing, object tracking, radar beamforming.



Stream processing in MATLAB

For developing efficient, readable stream processing programs in MATLAB, System objects:

- Process frames and then overwrite past frames with incoming data
- Initialize parameters only once as they are created
- Manage buffer updates, state updates, and indexing automatically, which speeds algorithm development
- Support MATLAB code generation and parallel computing workflows

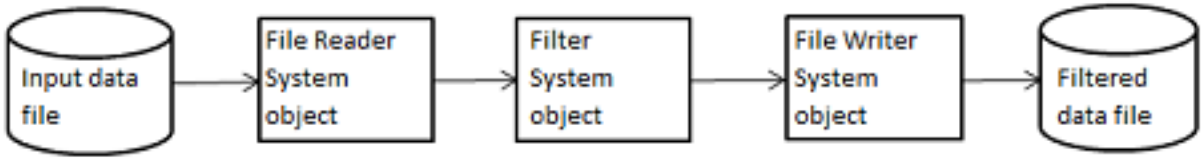
System objects are included in five MATLAB products:

- Audio System Toolbox
- DSP System Toolbox
- Communications System Toolbox
- Phased Array System Toolbox
- Computer Vision System Toolbox
- You can also define your own System objects to create new streaming algorithms

Audio System Toolbox	Communications System Toolbox	Computer Vision System Toolbox	DSP System Toolbox	Phased Array System Toolbox
...	...	...	...	...
audioDeviceReader	comm.TurboDecoder	vision.BlobAnalysis	dsp.BiquadFilter	phased.ArrayGain
audioDeviceWriter	comm.ViterbiDecoder	vision.EdgeDetector	dsp.LMSFiletr	phased.RadarTarget
audioOscillator	comm.MIMOChannel	vision.OptiocalFlow	dsp.LogicAnalyzer	phased.ULA
wavetableSynthesizer	comm.PSKModulator	vision.kalmanFilter	dsp.FIRDecimator	phased.URA
crossoverFilter	comm.CRCDetector	vision.VideoPlayer	dsp.CICDecimator	phased.Radiator
compressor	comm.PhaseNoise	vision.CornerDetector	dsp.FFT	phased.Transmitter
noiseGate	comm.LDPCEncoder	vision.HoughLines	dsp.SpectrumAnalyzer	phased.Platform
...	...	...	...	...

Using system objects example:

- Read and process large files
- File reader object
- Filter object
- File writer object



Example I: MATLAB Functions vs. System Objects

Try `stream_function.m`!

```
%% Locate source audio file.
fname = 'speech_dft_8kHz.wav';
type = 'Function'; % CHANGE THIS: 'Function' or 'SystemObject'

%% Obtain total number of samples and sampling rate from source file.
[y,Fs] = audioread(fname);
NrOfSamples  = length(y);

%% play sound:
sound(y,Fs);
disp('Playing audio! Press enter to continue!')
pause()

%% Define the filter to use.
b = fir1(160,.45);

%% Filter offline
y_filt = filter(b,1,y);
sound(y_filt,Fs);
disp('Playing filtered audio! Press enter to continue!')
pause()
```

Example I: MATLAB Functions vs. System Objects

Streaming with functions

```
%% Filter realtime with function or stream method
switch type
    case 'Function'
        % Initialize the filter states.
        z = zeros(1,numel(b)-1);
        % Define the amount of audio data to process at one time, and
             initialize the while loop index.
        frameSize = 1024;
        nIdx = 1;

        % Define a while loop to process the audio data.
        while nIdx <= NrOfSamples(1)-frameSize+1
            audio = audioread(fname,[nIdx nIdx+frameSize-1]);
            [y,z] = filter(b,1,audio,z);
            sound(y,Fs);
            pause(1024*1/Fs);
            nIdx = nIdx+frameSize;
        end
```

Example I: MATLAB Functions vs. System Objects

Streaming with system objects

```
    case 'SystemObject'
        % Define the System object to read the file.
        audioIn = dsp.AudioFileReader(fname,'OutputDataType','single');
        % Define the System object to filter the data.
        filtLP = dsp.FIRFilter('Numerator',fir1(160,.15));
        % Define the System object to play the filtered audio data.
        audioOut = audioDeviceWriter('SampleRate',audioIn.SampleRate);

        % Define the while loop to process the audio data.
        while ~isDone(audioIn)
            audio = audioIn();   % Read audio source file
            y = filtLP(audio);   % Filter the data
            audioOut(y);         % Play the filtered data
        end
end
```

Example II: Spectrum analyzer and playback

Try `realtime_audio.m`! Use various system objects for real time playback and spectrum analyzer from your microphone!

```
micReader         = audioDeviceReader;
speakerWriter     = audioDeviceWriter;
spectrumAnalyzer = dsp.SpectrumAnalyzer;
spectrumAnalyzer.SpectrumType = 'Spectrogram';
tic;
while (toc<30)
   y = micReader();
   spectrumAnalyzer(y);
   speakerWriter(y);
end
```

Table of contents

- A user defined MATLAB class can be easily created in a separate .m file.

```
classdef myClass
        properties
                ...
        end

        methods
                ...
        end
end
```

Investigate and try `myClass.m`

- Creating a MATLAB class Fraction which stores num and denum as private variables
- Create method which adds the two numbers.
- Create constructor, destructor and set function for the object!
- Try to access `a` and `b` directly!

```
>> Obj1 = myFrac(1,2); % Create object
>> Obj2 = myFrac(1,3); % Create object
>> sum = Obj1+Obj2
>> Obj1.setnum(4);        % Equivalent with seta(Obj,4)
>> double(Obj1)
```

```
classdef myFrac < handle
    properties (Access = private)
        num
        denum
    end
    methods
        %% C'tor
        function obj = myFrac(num, denum)
            if (nargin == 0)
                obj.num = 0;
                obj.denum = 1;
            elseif (nargin == 1)
                obj.num = num;
                obj.denum = 1;
            else
                obj.num = num;
                obj.denum = denum;
            end
        end
```

```
    function obj = setnum(obj,num)
        obj.num = num;
    end
    function obj = setdenum(obj,denum)
        obj.denum = denum;
    end
    %% Member function
    function disp(obj)
       disp(['a/b=',num2str(obj.num),'/',num2str(obj.denum)]);
    end
    %% Double conversion
    function d = double(obj)
        d = obj.num / obj.denum;
    end
    %% Adding two myFrac
    function obj_out = plus(o1, o2)
        num_tmp = o1.num*o2.denum + o2.num*o1.denum;
        denum_tmp = o1.denum * o2.denum;
        obj_out = myFrac( num_tmp,denum_tmp);
    end
end
```

Table of contents

Sending an email from MATLAB

Try `emailsending.m`!

```
% For GMAIL users: !!The following setting must be enabled !!
% Security -> Less secure apps & your Google Account -> Allow access
% https://myaccount.google.com/lesssecureapps
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parameters
emailfrom   = 'kollar.zsolt32@gmail.com'; % Your gmail email
password    = passcode(); % Password for account
smtpserver  = 'smtp.gmail.com';
port        = 465;
emailto     = 'kollar.zsolt32@gmail.com'; % Send to this email

%% Setup outgoing SMTP and email
setpref('Internet','SMTP_Server',smtpserver);
setpref('Internet','E_mail',emailfrom);
setpref('Internet','SMTP_Username',emailfrom);
setpref('Internet','SMTP_Password',password);
%% Set server specific ports and SSL authentication
props = java.lang.System.getProperties;
props.setProperty('mail.smtp.auth','true');
props.setProperty('mail.smtp.socketFactory.class',...
    'javax.net.ssl.SSLSocketFactory');
props.setProperty('mail.smtp.socketFactory.port',num2str(port));
```

Sending an email from MATLAB

```
%% Send email
disp('Sending email...')
try
    sendmail(emailto , 'Message title' , 'Hi Zsolt! This is MATLAB');
catch mex
    if strcmp(mex.identifier,'MATLAB:sendmail:SmtpError')
        disp('Wrong password or turn off firewall!');
    else
        disp('Unknow error');
    end
end
clear password
disp('Email sent!')
```

Table of contents

Problem of the week (POW): Poisoned barrels and fake coins

Problem 1: Suppose you have 100 barrels of wine, from which one of them is poisoned. One measurement takes 1 hour and you can only perform one measurement at once. How many hours does it take in worst case to find the poisoned barrel?

Problem 2: Suppose you have 9 coins that are identical in weight except one, which is lighter than the others, a fake. The difference is perceptible only by weighing them on balance scale – but only the coins themselves can be weighed. How can one isolate the fake coin with only two weighings?