Department of Computer Science III, University of Bonn
apl. Prof. Dr. Frank Kurth
Winter Term 2013/2014

Matlab Solutions

This document contains Matlab sample solutions for the exercises of Foundations of Audio Signal Processing.

Exercise 2.3(a)

```
function [ a, b ] = fasp_exe_2_3_a( r, phi )
%FASP_EXE_2_3_a Solves Exercise 2.3(a) of FASP 2013/14.
%   Conversion from polar to Cartesian coordinates.
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth

    a = r*cos(phi);
    b = r*sin(phi);

    figure
    hold on
    grid on

    % Unit circle for orientation:
    t = -2*pi:0.01:2*pi;
    plot(cos(t)+i*sin(t),':r')

    % Complex number.
    plot(a,b,'*b')

    legend('unit circle','complex number')

    % Axes:
    plot(0,-r-1:0.01:r+1,'--k')
    plot(-r-1:0.01:r+1,0,'--k')

    axis([-r-1,r+1,-r-1,r+1])
    xlabel('Real part')
    ylabel('Imaginary part')
    title(sprintf('Cartesian coordinates of %.2f+%.2fi = %.2fe^{%.2f*i}',a,

end
```

Exercise 2.3(b)

```
function fasp_exe_2_3_b( N )
%FASP_EXE_2_3_B Solves Exercise 2.3(b) of FASP 2013/14.
%   Plots the roots of unity.
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth

    figure
    hold on
    xlabel('Re')
    ylabel('Im')
    title(sprintf('Roots of unity for N = %u',N))

    % Circle:
    axis([-1.5,1.5,-1.5,1.5])
    t = -2*pi:0.001:2*pi;
    plot(cos(t)+i*sin(t),'r')

    % Axes:
    plot(-1.5:0.001:1.5,0,'k')
    plot(0,-1.5:0.001:1.5,'k')

    % Roots of unity:
    for k = 0:N-1
        z = exp(2*pi*1i*k/N);
        plot(real(z),imag(z),'*')
        text(real(z)*1.1,imag(z)*1.1,sprintf('k = %u',k))
    end
end
```

Exercise 2.3(c)

```
function fasp_exe_2_3_c
%FASP_EXE_2_3 Solves Exercise 2.3(c) of FASP 2013/14.
%   Shows that 'sin(alpha)' and '(exp(i*alpha)-exp(-i*alpha))/(2*i)' are eq
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth

    t = -3*pi:0.01:3*pi;
    sine = sin(t);
    diffexp = (exp(1i*t)-exp(-1i*t))/(2*1i);

    figure

    subplot(3,1,1)
    plot(t,sine)
    title('(i) sin(alpha)','Color','r','FontSize',11)
    xlabel('Time [seconds]')
    ylabel('Amplitude')

    subplot(3,1,2)
    plot(t,diffexp)
    title('(ii) (exp(i*alpha)-exp(-i*alpha))/(2*i)',...
        'Color','r','FontSize',11,'FontWeight','bold')
    xlabel('Time [seconds]')
    ylabel('Amplitude')

    subplot(3,1,3)
    plot(t,sine-diffexp)
    title('(iii) Difference of (i) and (ii)','Color','r','FontSize',11)
    xlabel('Time [seconds]')
    ylabel('Amplitude')

    axis tight
    linkaxes
end
```

Exercise 3.3

```
function fasp_exe_3_3(part)
%FASP_EXE_3_3 Solves Exercise 3.3 of FASP 2013/14.
%   Use 'b', 'c', or 'd' as an input.
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth

    N = 2;
    Fs = 8000;
    A = [0.5,0.25];
    phi = [0,0];

    str = sprintf('%s\n\n%s',...
        'Input missing or input not valid!','Give either ''b'',''c'', or ''
    if nargin < 1
        errordlg(str)
    else
        switch part
            case 'b'
                f = [2,16];
                sum_sine = sumsig( N, Fs, f, A, phi );

                figure
                ax(1) = subplot(2,1,1);
                plot(0:1/Fs:N,sum_sine)
                title(sprintf('Sum of two sine waves (2 and 16 Hz)'),...
                    'Color','r','FontSize',11,'FontWeight','bold')
                xlabel('Time [seconds]')
                ylabel('Amplitude')
                ax(2) = subplot(2,1,2);
                stem(0:1/Fs:N,sum_sine)
                xlabel('Time [seconds]')
                ylabel('Amplitude')
                linkaxes(ax,'x')
                axis tight
            case 'c'
                f = [200,440];
                sumsig( N, Fs, f, A, phi, 1 );
            case 'd'
                f = [200,440];
                sumsig( N, Fs, f, A, phi, 1 );
                [y,fs] = wavread(strcat(pwd,'\audio\sum_sine.wav'));
                sound(y,fs)
            otherwise
                errordlg(str)
        end
    end
end
```

```
function sum_sine = sumsig( N, Fs, f, A, phi, save )
%SUMSIG Generates a sum of sine waves
%   Input:  'N'    - signal length
%           'Fs'   - sampling frequency
%           'f'    - vector of frequencies
%           'A'    - vector of amplitudes
%           'phi'  - vector of phases
%           'save' - boolean for saving the signal in a '.wav' file
%   Output: 'sum_sine' - sum of sine waves
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth

    t = 0:1/Fs:N;
    sum_sine = zeros(1,length(t));
    for k = 1:length(f)
        sum_sine = sum_sine + A(k)*sin(2*pi*f(k)*t + phi(k));
    end

    if nargin < 6
        save = 0;
    end
    if save
        directory = strcat(pwd,'\audio\');
        if ~exist(directory,'dir')
            mkdir(directory)
        end
        wavwrite(sum_sine,Fs,strcat(directory,'sum_sine.wav'))
    end
end
```

Exercise 4.3

```
function fasp_exe_4_3
%FASP_EXE_4_3 Solves Exercise 4.3 of FASP 2013/14.
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth

    % PART A
    sampling_rate = 1000;
    t = -1:1/sampling_rate:1;
    sine40 = sin(2*pi*40*t);

    % PART B
    % calculate Fourier transform
    %fftshift shifts the zero-frequency component to center of the spectrum
    fft_sine40 = fftshift(fft(sine40));
    new_t = (-sampling_rate:sampling_rate)/(length(t)/sampling_rate);
    figure
    subplot(3, 1, 1)
    plot(t, sine40);
    title('Signal')
    subplot(3, 1, 2)
    plot(new_t, real(fft_sine40));
    title('real part')
    subplot(3, 1, 3)
    plot(new_t, imag(fft_sine40))
    title('imaginary part')

    % Fourier coeff
    %getFourierCoefficients(1000, 1000, 'sin', 40);

    % PART D
    sine80 = sin(2*pi*80*t);
    sum_sines = sine40 + sine80;
    fft_sum = fftshift(fft(sum_sines));
    figure
    subplot(3, 1, 1)
    plot(t, sum_sines);
    title('Signal')
    subplot(3, 1, 2)
    plot(new_t, real(fft_sum));
    title('real part')
    subplot(3, 1, 3)
    plot(new_t, imag(fft_sum))
    title('imaginary part')

    % PART E
    t = -1:1/sampling_rate:1.2;
    zeros_end = [sum_sines, zeros(1, 200)];
```

```
    fft_zeros_end = fftshift(fft(zeros_end, length(new_t)));
    %new_t = (-sampling_rate:sampling_rate)/(length(t)/sampling_rate);
    figure
    subplot(3, 2, 1)
    plot(t, zeros_end);
    title('Signal + zeros at the end')
    xlim([-1 1.2])
    subplot(3, 2, 3);
    plot(new_t, real(fft_zeros_end));
    title('real part')
    subplot(3, 2, 5);
    plot(new_t, real(fft_zeros_end));
    title('imaginary part')
    % now the zeros at the beginning
    t = -1.2:1/sampling_rate:1;
    zeros_start = [zeros(1, 200), sum_sines];
    fft_zeros_start = fftshift(fft(zeros_start, length(new_t)));
    subplot(3, 2, 2)
    plot(t, zeros_start);
    title('Signal + zeros at the begining')
    xlim([-1.2 1])
    subplot(3, 2, 4);
    plot(new_t, real(fft_zeros_start));
    title('real part')
    subplot(3, 2, 6);
    plot(new_t, real(fft_zeros_start));
    title('imaginary part')
    % ther peaks are at the same positions because the FT tells us which fr
    %are contained in the signal, but not when they are contained.

    % PART F
    sampling_rate = 8000;
    t = 0:1/sampling_rate:4;

    k = 150;
    chirp = sin(2*pi*k*t.^2);
    sound(chirp, sampling_rate);

    k = 300;
    chirp = sin(2*pi*k*t.^2);
    sound(chirp, sampling_rate);

    k = 50;
    chirp = sin(2*pi*k*t.^2);
    sound(chirp, sampling_rate);
end
```

Exercise 5.3

```
function fasp_exe_5_3(K,reps,p)
%FASP_EXE_5_3 Solves Exercise 5.3 of FASP 2013/14.
%   Performs an animated Fourier approximation of the step function.
%   Input:  'K'     - number of iterations
%           'reps'  - number of repetitions of the steps in the interval
%                     from '0' till 'reps'
%           'p'     - time interval between each plot specified in seconds
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth

    if nargin < 1
        K = 25;
        reps = 3;
        p = 0.1;
    end

    % Time interval:
    t = 0:0.001:1;

    % Prepare figure:
    figure
    title('Fourier approximation of the step function',...
        'Color','r','FontSize',11,'FontWeight','bold')
    xlabel('Time [seconds]')
    ylabel('Amplitude')
    hold on
    axis([-0.1,reps+0.1,-1.3,1.3])

    % Step function:
    step = step_fun(t);
    step = repmat(step,1,reps);

    % Fourier coefficients:
    t = 0:0.001:reps+(reps-1)*0.001;
    coeff = get_step_coeff(K);

    % Plot Fourier approximation:
    approx = zeros(1,length(t));
    for k = 1:K
        % Refine approximation:
        approx = approx + coeff(k)*sqrt(2)*sin(2*pi*k*t);
        % Plot refreshed data:
        cla
        plot(t,step)
        plot(t,approx,'r')
        text(0.2/reps,0.8,sprintf('k = %u',k),...
            'Units','normalized','Margin',5,...
```

```
            'Color','r','EdgeColor','k','BackgroundColor',.95*[1,1,1])
        leg = legend('step function','Fourier approximation');
        set(leg,'Location','SouthEast')
        pause(p)
    end
end

function step = step_fun(t)
%STEP_FUN Creates a step function in the interval 't' with
%   center 'c = (t(end)-t(1))/2', i.e.:
%
%                { -1,    if  t(1) <= t <= c
%   step(t)  =   {
%                {  1,    if  c < t <= t(end)

    step = ones(1,length(t));
    step(1:round(length(step)/2)) = -1;
end

function coeff = get_step_coeff(N)
%GET_STEP_COEFF Creates a vector of length 'N' containing the Fourier
%   coefficients of the step function for the interval '[0,1]', i.e.:
%
%   A_k = 0 for all k,
%
%           {  0,                 if  k  is even
%   B_k  =  {
%           {  -4*sgrt(2)/2*pi*k,  if  k  is odd

    indices = 1:N;
    coeff = zeros(1,length(indices));
    index = find(indices==0);
    if ~isempty(index)
        coeff(index) = 0;
        indices = setdiff(1:length(coeff),index);
    end
    odd = find(mod(indices,2)==1);
    coeff(odd) = -4*sqrt(2)/(2*pi)*(1./indices(odd));

end
```

Exercise 7.4

```
function fasp_exe_7_3(interval)
%FASP_EXE_7_3 Solves Exercise 7.3 of FASP 2013/14.
%   Performs a sinc approximation of a random signal given as a '.mat' file
%   with the name 'random_signal.mat' located in the current directory.
%   Input:  'interval' - section of the signal to approximate (1x2 vector)
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth

    d = load('random_signal.mat');

    N = length(d.signal)/d.Fs;
    t = (1:length(d.signal))/d.Fs;

    x = 1:N;
    if nargin < 1, interval = [1,N]; end

    figure('Name','Sinc approximation of a random signal')
    subplot(3,1,1)
    title('Random signal',...
        'Color','r','FontSize',11,'FontWeight','bold')
    hold on
    plot(t,d.signal,'linewidth',1.5)
    plot(x,d.signal(x*d.Fs),'ko','linewidth',1.5)
    xlabel('Time [seconds]')

    subplot(3,1,2)
    title('Approximation with sinc functions at each second',...
        'Color','r','FontSize',11,'FontWeight','bold')
    hold on
    plot(t,d.signal,'b--','linewidth',1.5)
    plot(x,d.signal(x*d.Fs),'ko','linewidth',1.5)
    % Sinc approximation:
    approx = zeros(1,length(d.signal));
    for k = interval(1):interval(end)
        plot(t,d.signal(k*d.Fs)*sin(pi*(t-k))./(pi*(t-k)),'r','linewidth',1
        approx = approx + d.signal(k*d.Fs)*sin(pi*(t-k))./(pi*(t-k));
    end
    xlabel('Time [seconds]')

    subplot(3,1,3)
    title('Signal and its sinc approximation',...
        'Color','r','FontSize',11,'FontWeight','bold')
    hold on
    plot(t,d.signal,'b--','linewidth',1.5)
    plot(t,approx,'r','linewidth',1.5)
    xlabel('Time [seconds]')
```

```
    linkaxes
end
```

Exercise 8.2(a)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   FASP_EXE_8_2_a Solves Exercise 8.2(a) of FASP 2013/14.
%
%   Applies an upsampling and a downsampling operator to a signal.
%   Uses the function 'operator.m'.
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Signal:
x = 1:11;

%-------------------------------------------------------------------------
% Sampling factors of part 8.2a (i):
M_up = 4;
M_down = 2;


% Preparation of plot:
figure('Name','FASP Exercise 8.2')
subplot(3,2,1:2)
stem(0:10,x,'filled')
xlabel('n')
ylabel('x(n)')
title('Original signal','Color','r','FontSize',11,'FontWeight','bold')


% Applying of the operators:
% First a downsampling on 'x' by 'M_down'...
x_down = operator( x, 'downsampling', M_down );
% ... then an upsampling by 'M_up' on the downsampled 'x':
x_up = operator( x_down, 'upsampling', M_up );



% Plot the results:
subplot(3,2,3)
stem(0:length(x_down)-1,x_down,'filled')
xlabel('n')
ylabel('x(n)')
title('$(\downarrow 2)[x](n)$',...
    'Interpreter','latex','Color','r','FontSize',11)

subplot(3,2,5)
```

```
stem(0:length(x_up)-1,x_up,'filled')
xlabel('n')
ylabel('x(n)')
title('$(\uparrow 4) \circ (\downarrow 2)[x](n)$',...
    'Interpreter','latex','Color','r','FontSize',11)


%------------------------------------------------------------------------
% Sampling factors of part 8.2a (ii):
M_up = 2;
M_down = 4;

% Applying of the operators:
% First an upsampling on 'x' by 'M_up'...
x_up = operator( x, 'upsampling', M_up );
% ... then a downsampling by 'M_down' on the upsampled 'x':
x_down = operator( x_up, 'downsampling', M_down );


% Plot the results:
subplot(3,2,4)
stem(0:length(x_up)-1,x_up,'filled')
xlabel('n')
ylabel('x(n)')
title('$(\uparrow 2)[x](n)$',...
    'Interpreter','latex','Color','r','FontSize',11)

subplot(3,2,6)
stem(0:length(x_down)-1,x_down,'filled')
xlabel('n')
ylabel('x(n)')
title('$(\downarrow 4) \circ (\uparrow 2)[x](n)$',...
    'Interpreter','latex','Color','r','FontSize',11)
```

```
function outSig = operator( inSig, type, param )
%OPERATOR Apllies the operator 'type' to the input signal 'inSig'.
%   The variable 'param' is used for the operator, e.g. the upsampling or
%   downsampling factor.
%   Input:  'inSig'  - input signal (given as a 1-dimensional vector)
%           'type'   - string: 'upsampling' or 'downsampling'
%           'param'  - operator parameter
%   Output: 'outSig' - output signal after having applied the operator


    switch type
        case 'downsampling'
            outSig = inSig(1:param:length(inSig));
        case 'upsampling'
            outSig = zeros(1, (length(inSig)-1)*param+1);
            inds = 1:param:length(outSig);
            outSig(inds) = inSig;
        otherwise
            errordlg(sprintf('Opeator ''%s'' unknown!',type))
            outSig = 'error';
    end
end
```

Exercise 8.2(b)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   FASP_EXE_8_2_b Solves Exercise 8.2(b) of FASP 2013/14.
%
%   Applies an upsampling and a downsampling operator to a signal.
%   Uses the function 'operator.m'.
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Read the signal 'spring.wav':
[x,fs] = wavread('spring.wav');


% Play the signal:
sound(x,fs)


% Sampling factors:
M_down = 4;
M_up = 4;


% Applying of the operators:
% First a downsampling on 'x' by 'M_down'...
x_down = operator( x, 'downsampling', M_down );
% ... then an upsampling by 'M_up' on the downsampled 'x':
x_up = operator( x_down, 'upsampling', M_up );


% Play the sampled signal:
% (fs = fs*M_up/M_down)
sound(x_up,fs)
```

Exercise 8.3

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   FASP_EXE_8_3 Solves Exercise 8.3 of FASP 2013/14.
%
%   Performs a uniform quantization of signal.
%   Uses the function 'quantizer.m'
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Generate a test signal:
t = 0:0.0001:1;
n = 5;
r = round(0.7*n*randn(1,n));
x = zeros(1,length(t));
for k = 1:n
    x = x + 0.3*k*sin(2*pi*t*r(k));
end


bits = [2,3,4];

figure('Name','FASP Exercise 8.3')
for k = 1:length(bits)
    % Perform quantization:
    outSig = quantizer( x, bits(k), [-3, 3] );

    % Plot original and quantized signal:
    subplot(length(bits),2,2*k-1)
    hold on
    plot(t,x)
    plot(t,outSig,'r')
    legend('Input','Quantized input')
    xlabel('time')
    title(sprintf('Quantization to %d bits',bits(k)),'Color','r','FontSize'

    % Plot quatization error:
    subplot(length(bits),2,2*k)
    hold on
    plot(t,20*log10(abs(x-outSig)));
    ylabel('dB')
    xlabel('time')
    title(sprintf('Quantization error (%d bits)',bits(k)),'Color','r','Font
end
```

```
function outSig = quantizer( inSig, bits, range )
%QUANTIZER Performs a linear quantization of 'inSig'.
%   Input:  'inSig'  - input signal
%           'bits'   - number of bits
%           'range'  - minimum and maxium value allowed
%   Output: 'outSig' - output signal


    if nargin < 3
        range = [min(inSig),max(inSig)];
    else
        if numel(range) == 1
            range = [-abs(range), abs(range)];
        end
    end
    if nargin < 2
        bits = 16;
    end

    % Find number of intervals:
    n = 2^bits - 1;

    % Normalize signal to range from 0 to 1:
    inNormalized = round((inSig-range(1))*n/(range(2)-range(1)));

    % Transform to range 'range' and
    % assign quantized value to each signal value:
    outSig = range(1)+inNormalized*(range(2)-range(1))/n;
end
```

Exercise 9.2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%    FASP_EXE_9_2 Solves Exercise 9.2 of FASP 2013/14.
%
%    Shows an animated plot of the convolution of two signals.
%    Uses the function 'convAnim.m'
%
%    Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Convolution of two rectangles
x = [1 1 1 1];
y = [1 1 1 1];
h = convAnim(x,y);
```

```
function h = convAnim(x,y)
%CONVANIM Produces an animated plot of the convolution of 'x' and 'y'.
%   The signal 'y' is flipped and moved step by step. Each time, the
%   pointwise product of 'x' and the shifted 'y' is calculated and summed
%   thus getting the k-th entry of the convolved signal 'h'.
%
%   Author: Alessia Cornaggia-Urrigshardt, apl. Prof. Dr. Frank Kurth


    % Determine the lengths of input signals and the convolved signal:
    lenx = length(x);
    leny = length(y);
    t = -leny+1:lenx+leny-2;
    lenh = length(t);
    h = zeros(1,lenh);

    % Flip signal 'y':
    y = fliplr(y);
    xp = zeros(1,lenh);
    xp(leny:leny+lenx-1) = x;

    figure
    for k = 1:lenx+leny-1
        % Plot x and the moved vesion of y:
        subplot(2,1,1);
        stem(0:lenx-1,x,'filled','b')
        xlim([t(1)-1,t(end)+1])
        hold on
        yp = zeros(1,lenh);
        yp(k:k+leny-1) = y;
        stem(t(k:k+leny-1),y,'filled','r')
        hold off
        legend('x(n)','y(n-k)',0);
        title('Signals x and y')
        xlabel('n')

        % Calculate index k of the convolved signal h:
        h(k) = sum(xp.*yp);

        subplot(2,1,2);
        stem(0:k-1,h(1:k),'filled','m')
        xlim([t(1)-1,t(end)+1])
        legend('h(n)')
        title(sprintf('Convolution of x and y at n = %u',k-1))
        xlabel('n')

        pause(0.5)
    end
    title('Convolution of x and y')
```

```
    h = h(1:lenx+leny-1);
end
```