Signals and Systems
Lab session 1

Cartesian and polar coordinates

As a starter exercise, you will implement your own versions of the built-in functions **cart2pol** and **pol2cart**. These functions allow you to switch between Cartesian coordinates (x, y) and polar coordinates (θ, r).

Exercise 1:
Implement the function **cart2polar**, which converts Cartesian coordinates (x, y) to polar coordinates (θ, r). The angle θ should be given relative to the positive x-axis. For example, the angle corresponding to Cartesian coordinates $(1, 0)$ is $\theta = 0$, while the angle corresponding to $(0, 1)$ is 1.57096 (numerical value of $\pi/2$).

The following skeleton code is already given, and can be found on Nestor in the file `ex1.m`. Complete the code, and test it. Once you are convinced that you solved the problem, you can submit the file to the automated judgment system *Themis* (link `themis.housing.rug.nl`). Make sure that you do not produce any extra output. Moreover, do not modify the `printf`-statement, since Themis compares your output with a reference output, which will fail if your output differs.

```
function [theta,r] = cart2polar(x,y)
% implement this function yourself
end

%%%%%%% main program starts here %%%%%%%
x = input("X=? ");
y = input("Y=? ");
[theta,r] = cart2polar(x,y);
printf("theta,r=%.3f,%.3f\n", theta, r);
```

Exercise 2:
Implement the function **polar2cart**, which converts polar coordinates (θ, r) to Cartesian coordinates (x, y). Once again, the angle θ is given relative to the positive x-axis. The following skeleton code is already given, and can be found on Nestor in the file `ex2.m`. Complete the code. Once you are convinced that you solved the problem, you can submit the file to *Themis*.

```
function [x,y] = polar2cart(theta, r)
% implement this function yourself
end

%%%%%%% main program starts here %%%%%%%
theta = input("theta=? ");
r = input("r=? ");
[x,y] = polar2cart(theta,r);
printf("x,y=%.3f,%.3f\n", x,y);
```

Sinusoids and samples

Exercise 3:
Implement the function **sinusoid2samples**, which produces a *sampled (i.e. discrete) signal* from a given sinusoid. The function takes as arguments amplitude A, frequency ω_0, phase φ, and f_s which is the sample rate (number of samples per second). The function **sinusoid2samples** should output the vector $x[k]$, where $x[k+1] = A\cos(\omega_0 * k/f_s + \varphi)$ for all $k \in [0..fs]$. Note that indexing in Octave(Matlab) starts with index 1.

The following skeleton code is already given, and can be found on Nestor in the file `ex3.m`. Complete the code, and submit the file to *Themis*.

```
function samples = sinusoid2samples(amplitude, omega, phi, samplerate)
% implement this function yourself
end

function printSamples(x)
  printf("[");
  nsamples = length(x);
  if nsamples > 0
    printf("%.3f", x(1));
    for i=2:nsamples
      printf(",%.3f", x(i));
    end
  end
  printf("]\n");
end

%%%%%%% main program starts here %%%%%%%
A   = input("A=? ");
f   = input("f(in Hz)=? ");
phi = input("phi(in radians)=? ");
fs  = input("samples per second=? ");

x = sinusoid2samples(A, 2*pi*f, phi, fs);

printSamples(x);

% uncomment the next line for a graphical plot,
% but comment it again before you submit it to themis!
%
% plot(x); pause;
```

Exercise 4:
Implement the function **samples2phase**, which takes as inputs the frequency ω_0, a sample vector $\mathbf{x} = (x_0, x_1, \ldots, x_n)$, and the corresponding sample rate f_s. This function should return the phase φ of the sampled sinusoid.

The following skeleton code is already given, and can be found on Nestor in the file `ex4.m`. Complete the code, and test it for several choices of the input parameters. You will likely find out that you need a high samplerate (> 1 kHz) to obtain reliable answers. Why do you think this is necessary? Answer this question in the file that you submit to Themis.

```
function phi=samples2phase(omega, x, samplerate)
% implement this function yourself
end

function samples = sinusoid2samples(amplitude, omega, phi, samplerate)
% put here the code from exercise 3
end

%%%%%%% main program starts here %%%%%%%
A   = input("A=? ");
f   = input("f(in Hz)=? ");
phi = input("phi(in radians)=? ");
fs  = input("samples per second=? ");

x = sinusoid2samples(A, 2*pi*f, phi, fs);
phi = samples2phase(2*pi*f, x, samplerate);
printf("phi=%.2f\n", phi);


%{
Answer here the posed question:
......
%}
```

Exercise 5:
Implement the function **samples2sinusoid**, which takes a sample vector x and a sample rate f_s. The vector x contains the sampling of 1 second of the sinusoid $A\cos(\omega t + \varphi)$. This function should return the amplitude A, frequency ω, and phase φ of the sinusoid.

The following skeleton code is already given, and can be found on Nestor in the file `ex5.m`. Complete the code, and test it for several choices of the input parameters. It is likely that you need an even higher samplerate than in the previous exercise to obtain reliable answers. Why do you think this is necessary? Which sampling rate is (at least) necessary to get reasonably reliable answers for frequencies $2\pi \leq \omega < 200\pi$? Answer these questions in the file that you submit to Themis.

```
function samples = sinusoid2samples(amplitude, omega, phi, samplerate)
% put here the code from exercise 3
end

function [amplitude,omega,phi]=samples2sinusoid(x, samplerate)
%implement this function yourself
end
```

```
%%%%%%% main program starts here %%%%%%%
A   = input("A=? ");
f   = input("f(in Hz)=? ");
phi = input("phi(in radians)=? ");
fs  = input("samples per second=? ");
x = sinusoid2samples(A, 2*pi*f, phi, fs);

[ampl, omega, phase]= samples2sinusoid(x, fs);
printf("x(t)=%.2f*cos(2*pi*%.2f*t + %.2f)\n", ampl, omega/(2*pi), phase);

%{
Answer here the posed questions:
.......
%}
```

Summing sinusoids

During the tutorials, we have seen several ways of summing to sinusoids with the same frequency. In this exercise, you will implement a function that can do the same. The function **addSinusoids** takes the parameters ω, A_0, φ_0, A_1, and φ_1, which represent the two sinusoids

$$\begin{aligned} z_0(t) &= A_0 \cos(\omega t + \varphi_0) \quad \text{and} \\ z_1(t) &= A_1 \cos(\omega t + \varphi_1) \end{aligned}$$

Your implementation of this function should return the frequency ω_2, amplitude A_2, and phase φ_2 of the sinusoid $z_2(t) = A_2 \cos(\omega_2 t + \varphi_2) = z_0(t) + z_1(t)$.

Exercise 6:
Implement the function **addSinusoids**. The following skeleton code is already given, and can be found on Nestor in the file `ex6.m`. Complete the code, and submit the file to *Themis*.

```
function [omega2,A2,phi2] = addSinusoids(omega, A0, phi0, A1, phi1)
% implement this function yourself
end

%%%%%%% main program starts here %%%%%%%
f    = input("frequency (in Hz)=? ");
A0   = input("A0=? ");
phi0 = input("phi0=? ");
A1   = input("A1=? ");
phi1 = input("phi1=? ");

[omega2,A2,phi2] = addSinusoids(f*2*pi, A0, phi0, A1, phi1);
printf("x(t)=%.2f*cos(2*pi*%.2f*t + %.2f)\n", A2, omega2/(2*pi), phi2);
```

Exercise 7:
For this exercise, we assume that the signals $z_0(t)$ and $z_1(t)$ are produced by sources that are located on a two-dimensional plane. Signal $z_0(t)$ is produced at coordinates (x_0, y_0), while signal

$z_1(t)$ is produced at coordinates (x_1, y_1). A microphone, located at coordinates (x_2, y_2), picks up the signals $z_0(t)$ and $z_1(t)$, which results in signal $z_2(t)$.

Implement the function **microphoneSignal** that calculates the parameters ω_2, A_2, and φ_2 of signal $z_2(t)$ given the ω, x_0, y_0, A_0, φ_0, x_1, y_1, A_1, φ_1, x_2, and y_2. The locations of the sources and microphone are given in meters, the speed of sound in this environment has been found to be 343 m/s.

The following skeleton code is already given, and can be found on Nestor in the file `ex7.m`. Complete the code, and submit the file to *Themis*.

```
function [omega2,A2,phi2] = microphoneSignal(omega,x0,y0,A0,phi0,x1,y1,A1,phi1,x2,y2)
% implement this function yourself
end

%%%%%%% main program starts here %%%%%%%

x0   = input("x0=? ");
y0   = input("y0=? ");
x1   = input("x1=? ");
y1   = input("y1=? ");
x2   = input("x2=? ");
y2   = input("y2=? ");

f    = input("frequency(in Hz)=? ");
A0   = input("A0=? ");
phi0 = input("phi0=? ");
A1   = input("A1=? ");
phi1 = input("phi1=? ");

[omega2,A2,phi2] = microphoneSignal(2*pi*f, x0, y0, A0, phi0, x1, y1, A1, phi1, x2, y2);
printf("x(t)=%.2f*cos(2*pi*%.2f*t + %.2f)\n", A2, omega2/(2*pi), phi2);
```