# Linear prediction

**Lab 13, SDP**

## Table of contents



# 1 Objective

Experiment with linear prediction for simple signals

# 2 Theoretical notions

Linear prediction represents the estimation of a sample of the signal $x[n]$ as a linear combination of $N$ previous samples:

$$x[n] \approx a_1 x[n-1] + a_2 x[n-2] + ... + a_N x[n-n]$$

Signals that approximately follow such a relationship are called "autoregressive" (AR). $N$ represents the order of the autoregressive model.

In Matlab, the function `lpc()` estimates the coefficients $a_k$ (refer to the documentation).

An alternative and more accurate method is provided in the function `lpc_exact()` along with the laboratory work.

# 3 Matlab examples

## 3.1 Generating some simple signals

Linear signal:

```
x = linspace(0,20,50);
plot(x)
```

Sinusoidal signal:

```
n = 0:50;
f = 0.1;
x = sin(2*pi*f*n);
plot(x)
```

Exponential signal $a^n u[n]$:

```
n = 0:20;
x = (1/2).^m;
plot(x)
```

Noise:

```
x = randn(1,100);   % or use rand()
plot(x)
```

### 3.2 Computing LPC coefficients

```
x = 1:1:10
order = 5;
a = lpc(x, order);
```

### 3.3 Prediction using the LPC coefficients

```
x = 1:1:500
order = 5;
a = lpc(x, order);

% Predict value at n=11 and append to x:
n=501;
x(n) = sum( x(n-1:-1:n-order) .* (-a(2:end))   )
```

## 4 Theoretical exercises

1. Consider the system described by the following equation:

$$y[n] = 0.8y[n-1] + x[n] + x[n-1],$$

   where $x[n]$ is a stationary random process with a mean of 0 and autocorrelation $\gamma_{xx}[m] = \left(\frac{1}{2}\right)^{|m|}$.

   a. Determine the power spectral density (PSD) of the output $y[n];
   b. Determine the autocorrelation function $\gamma_{yy}[m]$ of the output;
   c. Determine the variance $\sigma_y^2$ of the output.

## 5 Practical exercises

1. Linear prediction of a linear signal

   - Generate a linearly increasing signal with 200 samples, with constant slope $\Delta = 0.5$, starting from 5.

     Use `linspace()` or `start:step:stop`.

- Model the signal as an autoregressive AR(4) process. Compute the prediction coefficients $a_k$ using the Matkab function `lpc()`.
- Based on the prediction coefficients, using the prediction relation, predict the next 200 samples of the signal and append them.

  Plot the full resulting signal (400 samples).

  Use the relation: `sum(x(n-1:-1:n-ordin) .* (-a(2:end)))`
- Use the function `lpc_exact()` instead of `lpc()`. Is it better ?
- Change the AR order to AR(1), AR(2), AR(3), AR(10). What happens?

  What is the minimum order for which the prediction succeeds?

2. Linear prediction on various signals.

   Repeat the exercise above for the following singals:

   - Exponential signal: $x[n] = (0.9)^n u[n]$. Porniți de la un semnal de lungime 50, și estimați următoarele 50 eșantioane.
   - Sinusoidal signal: $x[n] = 3 \cdot \sin(2 * \pi * f * n)u[n], f = 0.05$. Porniți de la un semnal de lungime 50, și estimați următoarele 50 eșantioane.
   - Exponential sinusoidal: $x[n] = 0.8^n \cdot \sin(2 * \pi * f * n)u[n], f = 0.2$. Lungime 50 + 50
   - Damped sinusoidal signal: $x[n] = \frac{\sin(2*\pi*f*n)}{2*\pi*f*n}u[n], f = 0.05$. Porniți de la un semnal de lungime 50, și estimați următoarele 50 eșantioane.
   - Gaussian white noise (AWGN, generated with `randn()`). $x[n] = 0.8^n \cdot sin(2 * \pi * f * n)u[n], f = 0.2$. Lungime 500 + 100. Apoi lungime 20 + 100.
   - Uniform white noise (use `rand()`). $x[n] = 0.8^n \cdot sin(2 * \pi * f * n)u[n], f = 0.2$. Lungime 500 + 100. Apoi lungime 20 + 100.
   - Sinusoidal + white noise: $x[n] = 2 \cdot sin(2 * \pi * f * n)u[n] + AWGN, f = 0.05$. Lungime 100 + 100.
   - The `mtlb` signal. Estimate the next second of audio signal.
   - The first 150 samples from `mtlb`. Estimate the next second of audio signal.

3. Reducerea zgomotului prin predicție.

   Generați un semnal de forma:

   $x[n] = \sin(2 * \pi * f * n) + \text{zgomot alb}.$

Calculați coeficienții de predicție, și apoi estimați fiecare eșantion din semnalul $x[n]$ pe baza eșantioanelor precedente. Afișați semnalul astfel obținut ($x_2[n]$) cu semnalul original pe aceeași figură. Ce se observă?

4. Detecția vocii (Voice Activity Detector).

   - Încărcați semnalul audio `data_slow.wav` (cu `audioread()`), afișați-l grafic și redați-l audio.

   a. Utilizați funcția `buffer()` pentru a împărți semnalul în ferestre cu lungimea de aproximativ 25ms.
   b. Modelați fiecare segment semnalul ca un proces aleator AR(12), și găsiți coeficienții liniari de predicție pentru fiecare segment.
   c. Pentru fiecare segment, calculați energia coeficienților de predicție (suma coeficienților la pătrat). Afișați secvența de valori obținută.
   d. De pe grafic, alegeți un prag convenabil pentru a diferenția segmentele de voce de cele de pauză.
   e. Eliminați segmentele din semnal care sunt de pauză, și reuniți segmentele rămase întrun-un semnal întreg. Ascultați semnalul astfel obținut.

5. Repetați exercițiul anterior, dar adăugând peste semnalul inițial zgomot AWGN. Până la ce nivel de zgomot se obțin rezultate bune?

6. Incărcați imaginea `lena512.bmp`. Transformați în 0 valorile de pe liniile $20:30$, coloanele de la 100 la 150.

   Refaceți imaginea în felul următor: pentru fiecare linie separat, modelați primele 100 eșantioane cun un proces AR(10), apoi estimați cele 50 valori lipsa care urmează.

   Afișați imaginea astfel obținută.

# 6 Întrebări finale

1. TBD