Chap 9 : Adaptative Filtering

Ex 9.2 : Linear prediction using RLS on non stationary process

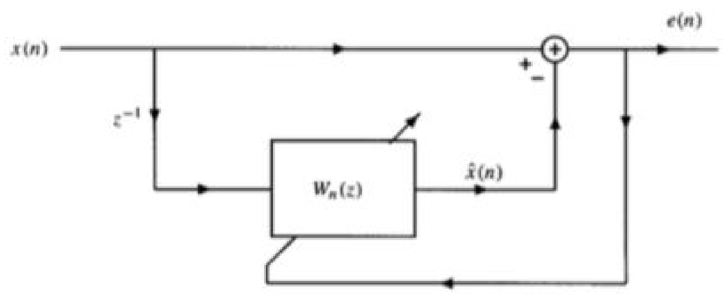
The goal of this exercice is to compare the LMS and the RLS algorithm for a one-step prediction of a non-stationary process.

$$x[n] = -a_n[1] \cdot \, x[n-1] - a_n[2] \cdot \, x[n-2] + v[n]$$

Where the coefficients $a_n[k]$ are changing accross time:

$$a_n[k] = \begin{cases} a_n[1] = -1.2728 \quad n \in [0, 100], & a_n[1] = 0 \quad n \in [101, 200] \\ a_n[2] = 0.81 \quad n \in [0, 100], & a_n[2] = 0.81 \quad n \in [100, 200] \end{cases}$$

To remind, the scheme of a one-step prediction is:



Therefore suppose we consider an adaptive linear predictor of the form:

$$\hat{x}[n] = w_n[1]x[n-1] + w_n[2]x[n-2]$$

The goal is to design the prediction using the two adaptive algorithms (RLS and LMS) and to compare the performance.

Data Generation

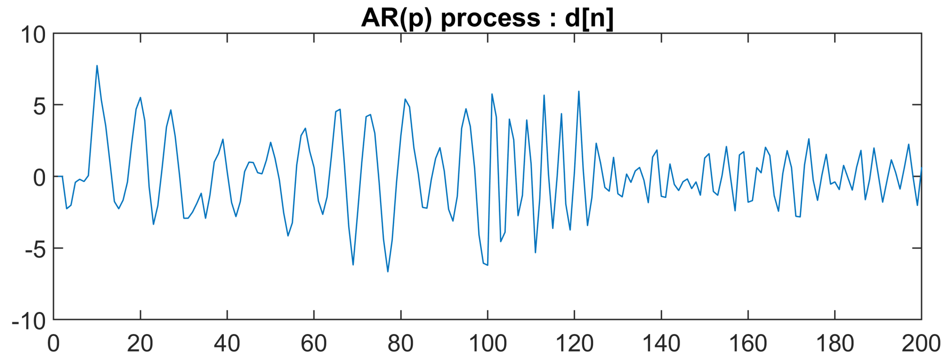
Parameter

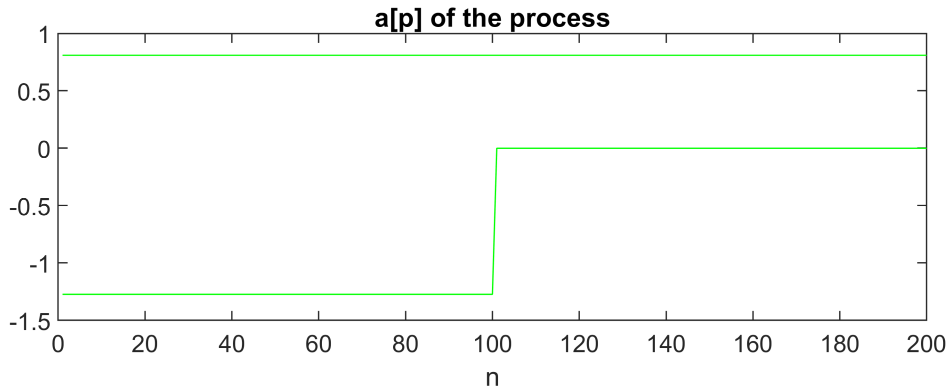
Let's set the parameters:

AFirst		ASecond		N	SigmaNoise
-1.2728	0.81	0	0.81	200	1

Generation

Let's generate the non-stationary process. The following is showing the signal and the coefficients accross time of the autoregressive process. You can directly load the signal **x**[n] from the data file as well the autoregressive coefficient accross time **A**.





Adaptive filter

Recursive autocorrelation

As seen in theory, the RLS algorithm can be expressed by computing the weighted deterministic autocorrelation matrix for x[n] and the deterministic cross-correlation between d[n] and x[n].

$$\boldsymbol{R}_{\boldsymbol{x}}(n)\boldsymbol{w}_{\boldsymbol{n}} = \boldsymbol{r}_{\mathbf{rdx}}(n)$$

$$\boldsymbol{w}_{\boldsymbol{n}} = \boldsymbol{R}_{\boldsymbol{x}}^{-1}(n) \cdot \boldsymbol{r}_{\mathbf{rdx}}(n)$$

The determnistic autocorrelation is computed :

$$\boldsymbol{R}_{\boldsymbol{x}}(n) = \sum_{i=0}^{n} \lambda^{n-i} \boldsymbol{x}^*[i]\boldsymbol{x}^T[i] = \lambda \boldsymbol{R}_{\boldsymbol{x}}(n-1) + \boldsymbol{x}^*[i]\boldsymbol{x}^T[i]$$

And the determinstic crosscorrelation is computed:

$$\boldsymbol{r}_{\mathbf{rdx}}(n) = \sum_{i=0}^{n} \lambda^{n-i} d[i]\boldsymbol{x}^*[i] = \lambda\, \boldsymbol{r}_{\mathbf{rdx}}(n-1) + d[i]\boldsymbol{x}^*[i]$$

Where $\boldsymbol{x}[i]$ is the sample vector :

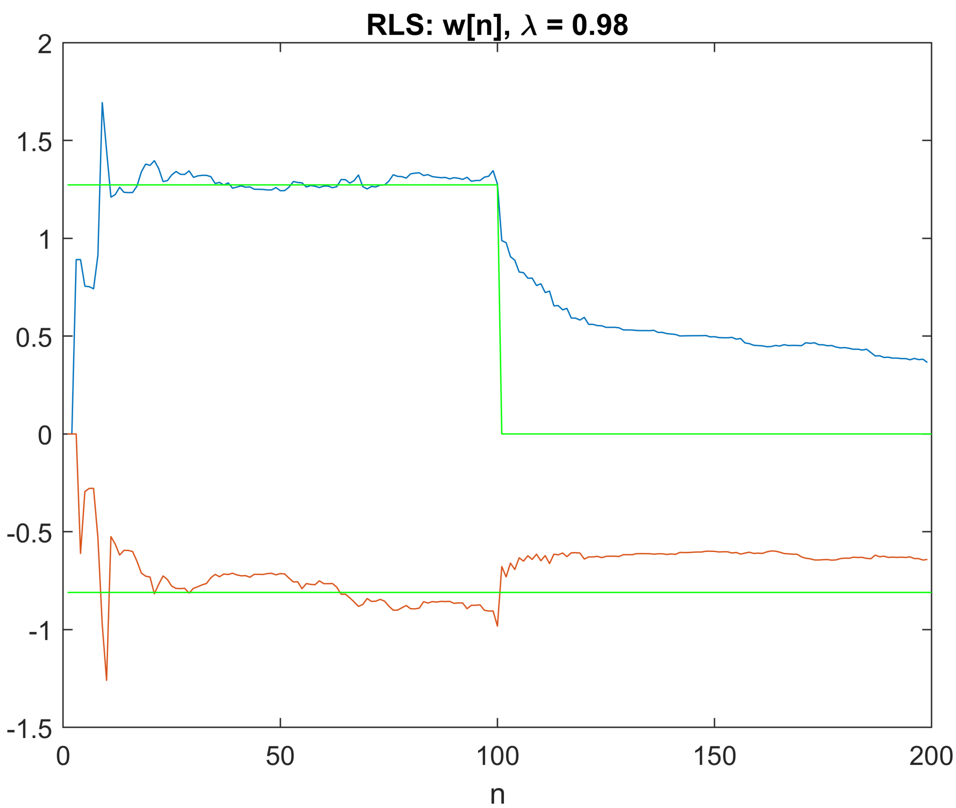
$$\mathbf{x}[i] = \begin{bmatrix} x[i] \\ x[i-1] \\ \vdots \\ x[i-p] \end{bmatrix}$$

And for a predictive filter, the desired output $d[i]$ is simply the futur sample :

$$d[i] \ = \ x[i+1]$$

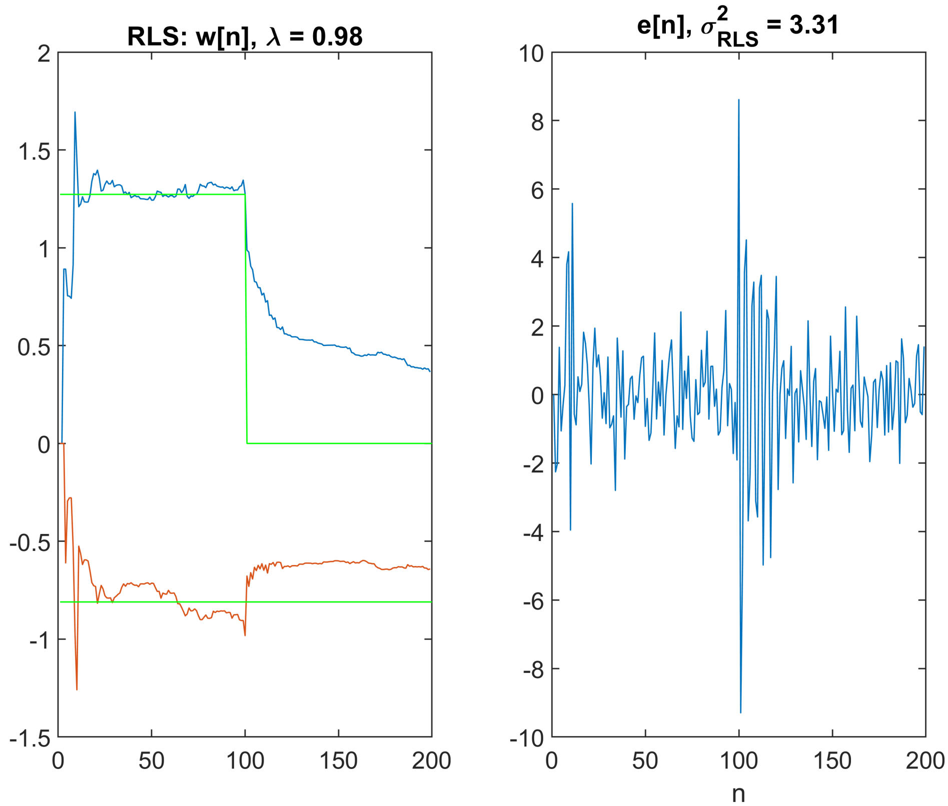
Let's start to compute the filter coefficients **w** using this recursive autocorrelation. For each sample n, a new vector of coefficient **w** is computed. The number of coefficient must be equal to nc = 2 and $\lambda = 0.98$. The vector **w** can be saved in a matrix **W**(n,:) of preinitialize **W** = zeros(N-1,nc) where N is the total number of sample.

Compare the computed coefficients **W** to the optimum coefficients which are simply the $w_{\text{nOptim}}[k] \ = \ -a_n[k]$



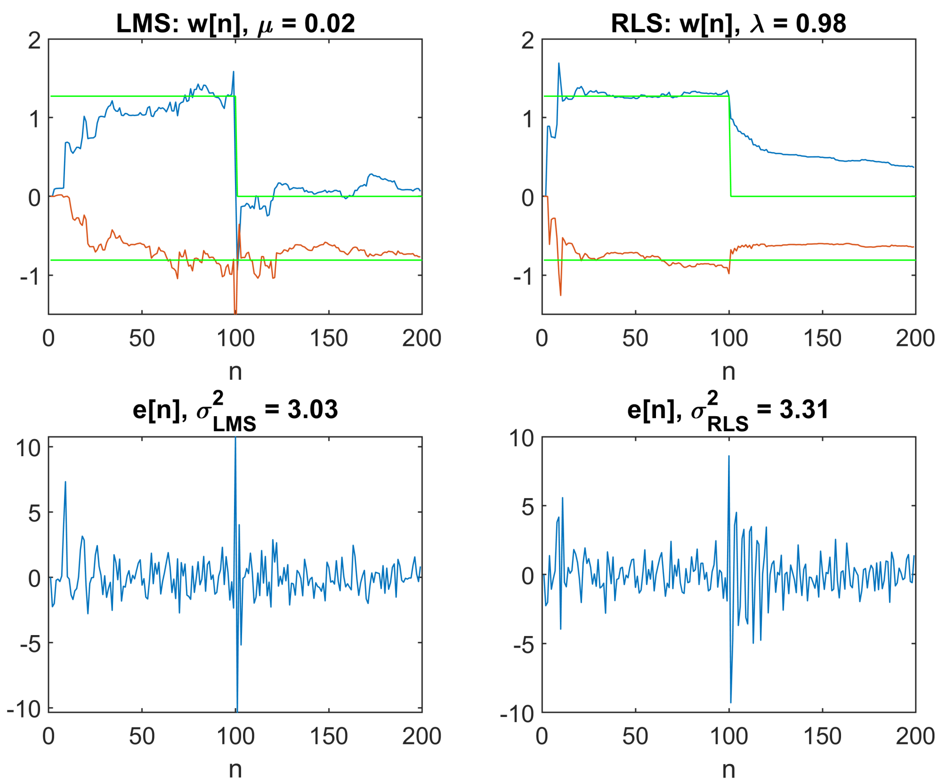
Design RLS

To avoid this inverse of $\boldsymbol{R}_x(n)$ accross time, we could compute the inverse recursively as well. For that, you can use the RLS calculation given in the theory and implement the function myRLS(). Then, you can compare the found coefficients to the optimum coefficients. Use as previously nc = 2 and $\lambda = 0.98$.



Compare to LMS

Finally, compare the coefficients with the LMS algorithm by using $\mu = 0.02$.


LMS: w[n], μ = 0.02
RLS: w[n], λ = 0.98
e[n], σ²LMS = 3.03
e[n], σ²RLS = 3.31
n