

MATH 545: INTRODUCTION TO TIME SERIES ANALYSIS

WINTER 2025

ALGORITHMS FOR FORECASTING

Recall that the optimal linear forecast for a stationary process is found by solving a least-squares problem: we minimize the mean square (forecast or prediction) error

$$\text{MSE}_n(h) = \mathbb{E} \left[\left(X_{n+h} - \hat{X}_{n+h} \right)^2 \right] \quad (1)$$

for a specified n and $h \geq 1$. We first note the decomposition

$$\begin{aligned} \mathbb{E} \left[\left(X_{n+h} - \hat{X}_{n+h} \right)^2 \right] &= \mathbb{E} \left[\left(X_{n+h} - m_n(h) + m_n(h) - \hat{X}_{n+h} \right)^2 \right] \\ &= \mathbb{E} \left[(X_{n+h} - m_n(h))^2 \right] + 2\mathbb{E} \left[(X_{n+h} - m_n(h))(m_n(h) - \hat{X}_{n+h}) \right] + \mathbb{E} \left[(m_n(h) - \hat{X}_{n+h})^2 \right] \end{aligned} \quad (2)$$

where $m_n(h) = \mathbb{E}[X_{n+h}|X_1, \dots, X_n]$ is the conditional expectation of X_{n+h} given $X_1, \dots, X_n$. In these three terms, the first is some constant that does not depend on the choice of $\hat{X}_{n+h}$; the second is zero, as, by iterated expectation

$$\mathbb{E} \left[(X_{n+h} - m_n(h))(m_n(h) - \hat{X}_{n+h}) \right] = \mathbb{E}_{X_{1:n}} \left[\mathbb{E}_{X_{n+h}|X_{1:n}} \left[(X_{n+h} - m_n(h))(m_n(h) - \hat{X}_{n+h}) | X_{1:n} \right] \right]$$

if $X_{1:n} \equiv (X_1, \dots, X_n)$ – for the inner expectation, we have

$$\mathbb{E}_{X_{n+h}|X_{1:n}} \left[(X_{n+h} - m_n(h))(m_n(h) - \hat{X}_{n+h}) | X_{1:n} \right] = (m_n(h) - \hat{X}_{n+h}) \mathbb{E}_{X_{n+h}|X_{1:n}} [(X_{n+h} - m_n(h)) | X_{1:n}]$$

as both $m_n(h)$ and $\hat{X}_{n+h}$ are functions of $X_{1:n}$, and $\mathbb{E}_{X_{n+h}|X_{1:n}} [(X_{n+h} - m_n(h)) | X_{1:n}] = 0$. Therefore we minimize (1) by minimizing the third term in (2): this can be achieved by setting

$$\hat{X}_{n+h} = m_n(h) = \mathbb{E}[X_{n+h}|X_1, \dots, X_n]$$

in which case

$$\text{MSE}_n(h) = \mathbb{E} \left[(X_{n+h} - \mathbb{E}[X_{n+h}|X_1, \dots, X_n])^2 \right]. \quad (3)$$

Using a linear representation for $m_n(h)$, we can get a more explicit result

$$\hat{X}_{n+h} = a_0 + \sum_{i=1}^n a_i X_{n-i+1}.$$

This reduces to choosing the coefficients $a = (a_1, \dots, a_n)^\top$. We minimize using calculus: we have

$$\frac{\partial \text{MSE}_n(h)}{\partial a_0} = \mathbb{E} \left[\left(X_{n+h} - a_0 - \sum_{i=1}^n a_i X_{n-i+1} \right) \right] \quad (4)$$

$$\frac{\partial \text{MSE}_n(h)}{\partial a_j} = \mathbb{E} \left[\left(X_{n+h} - a_0 - \sum_{i=1}^n a_i X_{n-i+1} \right) X_{n-j+1} \right], j = 1, \dots, n. \quad (5)$$

allowing for differentiation under the expectation. We must solve after setting (4) and (5) equal to zero simultaneously. From equation (4), we have that

$$\mathbb{E}[X_{n+h}] - a_0 - \sum_{i=1}^n a_i \mathbb{E}[X_{n-i+1}] = 0$$

and if $\{X_t\}$ is stationary with mean μ , we may conclude that the optimal choice must satisfy

$$a_0 = \mu \left(1 - \sum_{i=1}^n a_i \right).$$

From (5), we have for each $j = 1, \dots, n$,

$$\mathbb{E}[X_{n+h}X_{n-j+1}] - a_0\mathbb{E}[X_{n-j+1}] - \sum_{i=1}^n a_i\mathbb{E}[X_{n-i+1}X_{n-j+1}] = 0.$$

Noting that from the earlier result, we have

$$a_0\mathbb{E}[X_{n-j+1}] = \mu^2 \left(1 - \sum_{i=1}^n a_i \right)$$

and hence, rearranging

$$\mathbb{E}[X_{n+h}X_{n-j+1}] - \mu^2 - \sum_{i=1}^n a_i(\mathbb{E}[X_{n-i+1}X_{n-j+1}] - \mu^2) = 0.$$

so that

$$\sum_{i=1}^n a_i \text{Cov}[X_{n-i+1}, X_{n-j+1}] = \text{Cov}[X_{n+h}, X_{n-j+1}]$$

or equivalently

$$\sum_{i=1}^n a_i \gamma_X(i-j) = \gamma_X(j+h-1) \quad j = 1, \dots, n.$$

Stacking these n equations, we are left with

$$\begin{bmatrix} \gamma_X(0) & \gamma_X(1) & \gamma_X(2) & \cdots & \gamma_X(n-1) \\ \gamma_X(1) & \gamma_X(0) & \gamma_X(1) & \cdots & \gamma_X(n-2) \\ \gamma_X(2) & \gamma_X(1) & \gamma_X(0) & \cdots & \gamma_X(n-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_X(n-1) & \gamma_X(n-2) & \gamma_X(n-3) & \cdots & \gamma_X(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \gamma_X(h) \\ \gamma_X(h+1) \\ \gamma_X(h+2) \\ \vdots \\ \gamma_X(h+n-1) \end{bmatrix}$$

or equivalently

$$\mathbf{\Gamma}_X(n)\mathbf{a} = \mathbf{\Gamma}_X(n, h) \tag{6}$$

say, where

$$\mathbf{\Gamma}_X(n, h) = (\gamma_X(h), \dots, \gamma_X(h+n-1))^\top.$$

Thus provided $\mathbf{\Gamma}_X(n)$ is non-singular, we have that the optimal $\mathbf{a}$ is the solution to (6)

$$\mathbf{a}(n, h) = \{\mathbf{\Gamma}_X(n)\}^{-1}\mathbf{\Gamma}_X(n, h)$$

which requires us to compute $\{\mathbf{\Gamma}_X(n)\}^{-1}$.

The Levinson-Durbin Algorithm: Without loss of generality, take $\mathbb{E}[X_t] = 0$. Consider forecasting for $h = 1$, and write

$$\hat{X}_{n+1} = \sum_{i=1}^n \phi_{n,i} X_{n-i+1} = \phi_n^\top X_{n:1}$$

where $\phi_n = (\phi_{n,1}, \dots, \phi_{n,n})^\top$, which corresponds in our prediction forecast to $\mathbf{a}(n, 1)$. The optimal choice solves the linear system

$$\mathbf{\Gamma}_X(n) \phi_n = \mathbf{\Gamma}_X(n, 1) = \gamma_n^R,$$

say, which is an $n \times 1$ system of equations, yielding minimum MSE

$$v_n = \gamma_X(0) - \phi_n^\top \gamma_n^R$$

The Levinson-Durbin recursion solves the equation without matrix inversion, in order n^2 operations (most matrix inversion procedures are order n^3) by exploiting the Toeplitz structure of $\mathbf{\Gamma}_X(n)$. One can also use singular value, eigenvalue decomposition or Cholesky decomposition; these methods have special implementation for positive-definite Toeplitz structure matrices.

Initialization: For $n = 1$, set

$$\phi_{1,1} = \frac{\gamma_X(1)}{\gamma_X(0)} = \rho_X(1) \quad v_0 = \gamma_X(0) \quad v_1 = \gamma_X(0)(1 - \phi_{1,1}^2).$$

Recursion: Define ϕ_n in terms of ϕ_{n-1} and v_{n-1} as follows:

$$\phi_{n,n} = \frac{1}{v_{n-1}} \left[\gamma_X(n) - \phi_{n-1}^\top \gamma_{n-1}^R \right]$$

where $\mathbf{\Gamma}_X^R(n-1, 1) = (\gamma_X(n-1), \gamma_X(n-1), \dots, \gamma_X(1))^\top$ (the superscript 'R' means 'reverse') and

$$\phi_{n,1:(n-1)} = \phi_{n-1} - \phi_{n,n} \phi_{n-1}^R$$

where ϕ_{n-1} and ϕ_{n-1}^R are $(n-1) \times 1$ vectors with $\phi_{n-1}^R = (\phi_{n-1,n-1}, \phi_{n-1,n-2}, \dots, \phi_{n-1,1})^\top$, and $\phi_{n,n}$ is a scalar. Then set

$$v_n = v_{n-1}(1 - \phi_{n,n}^2).$$

Proof

To see why this works, let

$$\mathbf{R}_n = \frac{1}{\gamma_X(0)} \mathbf{\Gamma}_X(n)$$

be the autocorrelation matrix and let

$$\boldsymbol{\rho}_n = \frac{1}{\gamma_X(0)} (\gamma_X(1), \dots, \gamma_X(n))^\top \quad \boldsymbol{\rho}_n^R = (\rho_X(n), \dots, \rho_X(1))^\top.$$

The recursion computes the inverse of $\mathbf{P}_{n+1}$ from the inverse of $\mathbf{P}_n$ using rank-one operations. Note first that

$$\mathbf{\Gamma}_X(n) \phi_n = \gamma_n^R \implies \mathbf{R}_n \phi_n = \boldsymbol{\rho}_n.$$

We need to verify that

$$\mathbf{R}_n \phi_n = \boldsymbol{\rho}_n \implies \mathbf{R}_{n+1} \phi_{n+1} = \boldsymbol{\rho}_{n+1}.$$

For $n = 1$, $\mathbf{R}_1 = 1$, $\phi_1 = \phi_{1,1} = \rho(1) = \rho_{1,1}$. We establish the result by induction on n . Assume that $\mathbf{R}_k \phi_k = \boldsymbol{\rho}_k$. Now

$$\mathbf{R}_{k+1} = \begin{bmatrix} \mathbf{R}_k & \boldsymbol{\rho}_k^R \\ \{\boldsymbol{\rho}_k^R\}^\top & 1 \end{bmatrix}$$

and $\mathbf{R}_k \phi_k = \rho_k$, therefore

$$\mathbf{R}_k \phi_k^R = \rho_k^R$$

using the symmetry of $\mathbf{R}_k$. In the proposed recursion, we have that

$$\mathbf{R}_{k+1} \phi_{k+1} = \begin{bmatrix} \mathbf{R}_k & \rho_k^R \\ \{\rho_k^R\}^\top & 1 \end{bmatrix} \begin{bmatrix} \phi_k - \phi_{k+1,k+1} \phi_k^R \\ \phi_{k+1,k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$$

where $\mathbf{a}$ is equal to a $k \times 1$ system of the form

$$\mathbf{a} = \mathbf{R}_k \phi_k - \phi_{k+1,k+1} \mathbf{R}_k \phi_k^R + \phi_{k+1,k+1} \rho_k = \mathbf{R}_k \phi_k = \rho_k$$

by the inductive hypothesis. For $\mathbf{b}$, we have

$$\mathbf{b} = \{\rho_k^R\}^\top \phi_k - \phi_{k+1,k+1} \{\rho_k^R\}^\top \phi_k^R + \phi_{k+1,k+1} = \{\rho_k^R\}^\top \phi_k - \phi_{k+1,k+1} (1 - \{\rho_k^R\}^\top \phi_k^R)$$

We have now

$$\phi_{k+1,k+1} = \frac{1}{v_k} \left[\gamma_X(k+1) - \phi_k^\top \gamma_k^R \right]$$

where

$$v_k = \gamma_X(0) - \phi_k^\top \mathbf{\Gamma}_X(k, 1) = \gamma_X(0) (1 - \phi_k^\top \rho_k) = \gamma_X(0) (1 - \{\rho_k^R\}^\top \phi_k^R)$$

From the above formula, we have

$$\phi_{k+1,k+1} = \frac{1}{\gamma_X(0) (1 - \{\rho_k^R\}^\top \phi_k^R)} \left[\gamma_X(k+1) - \phi_k^\top \gamma_k^R \right] \quad (7)$$

changing the order of the inner product and from $\mathbf{b}$, we have

$$\begin{aligned} \rho_k^\top \phi_k + \frac{\left[\gamma_X(k+1) - \phi_k^\top \gamma_k^R \right]}{\gamma_X(0) (1 - \rho_k^\top \phi_k^R)} (1 - \rho_k^\top \phi_k^R) &= \{\rho_k^R\}^\top \phi_k + \rho_X(k+1) - \phi_k^\top \rho_k^R \\ &= \rho_X(k+1) \end{aligned}$$

as required. Therefore the required recursion is valid by the inductive argument. ■

We have effectively performed a matrix inversion by considering a block decomposition

$$\mathbf{R}_{k+1} = \begin{bmatrix} \mathbf{R}_k & \rho_k^R \\ \{\rho_k^R\}^\top & 1 \end{bmatrix}$$

and then computed $\mathbf{R}_{k+1}^{-1}$ using $\mathbf{R}_k^{-1}$. This works because of the nature of the matrix (Toeplitz structure).

For v_n , we have

$$\begin{aligned} v_n &= \mathbb{E}[(X_{n+1} - \phi_n^\top X_{n:1})^2] = \gamma_X(0) - \phi_n^\top \gamma_{n-1}^R \\ &= \gamma_X(0) - \phi_{n-1}^\top \gamma_{n-1}^R + \phi_{n,n} \{\phi_{n-1}^R\}^\top \gamma_{n-1}^R - \phi_{n,n} \gamma_X(n) \\ &= v_{n-1} + \phi_{n,n} (\{\phi_{n-1}^R\}^\top \gamma_{n-1}^R - \gamma_X(n)) \\ &= v_{n-1} - \phi_{n,n}^2 (\gamma_X(0) - \{\phi_{n-1}^R\}^\top \gamma_{n-1}^R) \\ &= v_{n-1} - \phi_{n,n}^2 v_{n-1} \\ &= v_{n-1} (1 - \phi_{n,n}^2) \end{aligned}$$

plugging in (7). That is, $v_n = v_{n-1} (1 - \phi_{n,n}^2)$ and as $v_n \geq 0$, imply $\phi_{n,n}^2 < 1$, $|\phi_{n,n}| < 1$ thus $v_n \leq v_{n-1}$, that is, the $\{v_n\}$ form a non-increasing sequence.

Partial autocorrelation: Using the Levinson-Durbin algorithm, define the function $\alpha_X(h)$ by

$$\alpha_X(h) = \begin{cases} 1 & \text{if } h = 0 \\ \phi_{h,h} & \text{if } h \geq 1 \end{cases}$$

for $h = 0, 1, 2, \dots$. This $\alpha_X(h)$ is the *partial autocorrelation function* (PACF). For random variables X, Y, Z , the *partial correlation* is defined by

$$\text{Corr}[(X - \mathbb{E}[X|Z]), (Y - \mathbb{E}[Y|Z])];$$

this is the correlation between residuals obtained by regressing in turn X and on Z and Y on Z . The PACF computes the autocorrelation between

$$X_t - \mathbb{E}[X_t|X_{(t+1):(t+h-1)}] \quad \text{and} \quad X_{t+h} - \mathbb{E}[X_{t+h}|X_{(t+1):(t+h-1)}].$$

Consider a causal stationary AR(p) process; for $h > p$,

$$\mathbb{E}[X_{t+h}|X_{(t+1):(t+h-1)}] = \phi_1 X_{t+h-1} + \dots + \phi_p X_{t+h-p} \quad \mathbb{E}[X_t|X_{(t+1):(t+h-1)}] = f(X_{t+1}, \dots, X_{t+h-1})$$

for some function $f(\cdot)$, where by definition $X_{t+h} - \mathbb{E}[X_{t+h}|X_{(t+1):(t+h-1)}] = Z_{t+h}$, therefore

$$\text{Corr}[X_t - \mathbb{E}[X_t|X_{(t+1):(t+h-1)}], X_{t+h} - \mathbb{E}[X_{t+h}|X_{(t+1):(t+h-1)}]] = \text{Corr}[X_t - \mathbb{E}[X_t|X_{(t+1):(t+h-1)}], Z_{t+h}] = 0$$

as $\{X_t\}$ is causal so $X_t - \mathbb{E}[X_t|X_{(t+1):(t+h-1)}]$ depends on $\{Z_s, s \leq t+h-1\}$ only. Thus $\alpha_X(h) = 0$ for $h > p$, while for $h \leq p$, $\alpha_X(h) \neq 0$. That is, we can diagnose an AR(p) structure by inspecting whether the partial autocorrelation drops to zero at some finite lag.

EXAMPLE Consider the AR(3) process

$$X_t = -1.4X_{t-1} - 0.48X_{t-2} + 0.16X_{t-3} + Z_t$$

or equivalently

$$(1 + 1.4B + 0.48B^2 - 0.16B^3)X_t = Z_t.$$

or $\Phi(B)X_t = Z_t$ with $\Phi(z) = (1 + 1.4z + 0.48z^2 - 0.16z^3)$. We have the factorization

$$\Phi(z) = (1 - 0.2z)(1 + 1.6z + 0.8z^2) = (1 - \xi_1 z)(1 - \xi_2 z)(1 - \xi_3 z)$$

and first reciprocal root is $\xi_1 = 0.2$. The two remaining roots are

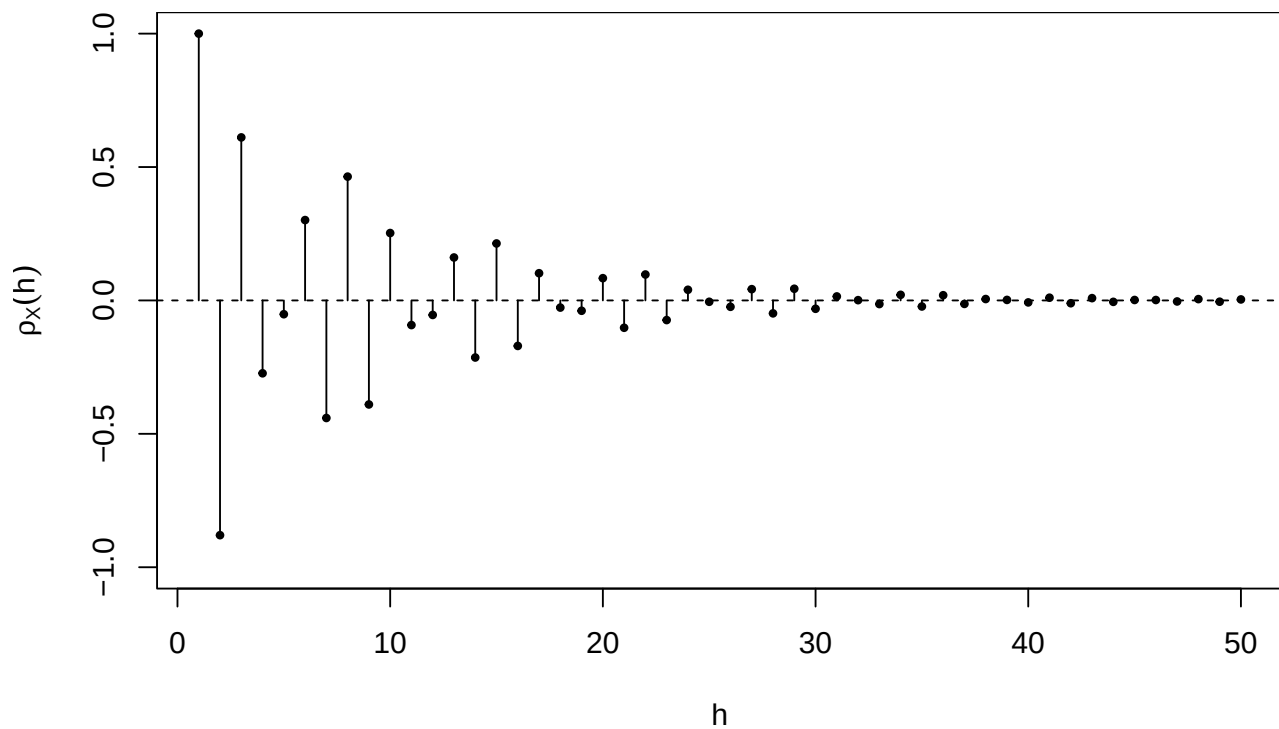
$$\frac{-1.6 \pm \sqrt{2.56 - 3.2}}{2} = \frac{-1.6 \pm \sqrt{-0.64}}{2} = -0.8 \pm 0.4i$$

so the reciprocal roots are $\xi_2 = -0.8 + 0.4i$ and $\xi_3 = -0.8 - 0.4i$. All the reciprocal roots are less than one in modulus, so the process is stationary and causal.

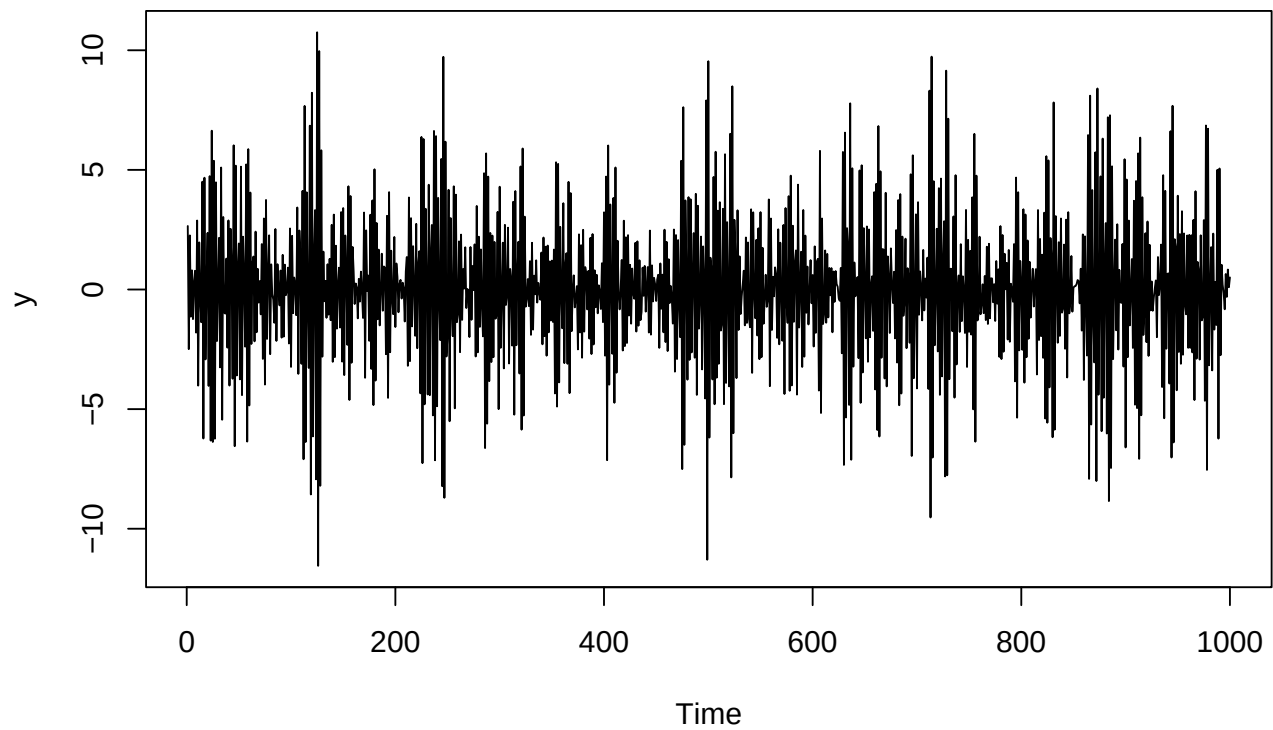
```
alpha.vals<-c(0.2,-0.8+0.4i,-0.8-0.4i)
ar.coefs<-rep(1,length(alpha.vals)+1)
for(j in 1:length(alpha.vals)){
  ar.coefs.old<-ar.coefs
  if(j > 1){
    for(k in 2:j){
      ar.coefs[k]<-ar.coefs.old[k]-alpha.vals[j]*ar.coefs.old[k-1]
    }
  }
  ar.coefs[j+1]<-alpha.vals[j]*ar.coefs.old[j]
}
ar.coefs #These are the AR coefficients

+ [1] 1.00+0i 1.40+0i 0.48+0i -0.16+0i

n<-200
rho.fn1<-ARMAacf(ar=Re(-ar.coefs[2:4]), lag.max = n+1) #Simulate data
par(mar=c(4,4,0,1))
plot(rho.fn1[1:50],type="n",ylim=range(-1,1),xlab="h",ylab=expression(rho[X](h)))
abline(h=0,lty=2)
for(h in 1:50){lines(c(h,h),c(0,rho.fn1[h]))}
points(rho.fn1[1:50],pch=19,cex=0.5)
```



```
set.seed(26)
y<-arima.sim(n=1000,model=list(ar=Re(-ar.coefs[2:4])))
par(mar=c(4,4,0,1))
plot(y)
```



```
#####
#Levinson-Durbin
psi.Vals<-ARMAtoMA(ar=Re(-ar.coefs[2:4]),lag.max=10000)
Variance<-1+sum(psi.Vals^2)
```

```

gamma.fn1<-Variance*rho.fn1[-1]
Phi.mat<-matrix(0,nrow=n,ncol=n)
v<-rep(0,n+1)
v[0+1]<-Variance
Phi.mat[1,1]<-gamma.fn1[1]/Variance #Initialize
v[1+1]<-Variance*(1-Phi.mat[1,1]^2)
for(j in 2:n){ #Recursion
  Phi.mat[j,j]<-(gamma.fn1[j]-sum(Phi.mat[j-1,1:(j-1)]*rev(gamma.fn1[1:(j-1)])))/v[(j-1)+1]
  Phi.mat[j,1:(j-1)]<- Phi.mat[j-1,1:(j-1)] - Phi.mat[j,j] * rev(Phi.mat[j-1,1:(j-1)])
  v[j+1]<-v[(j-1)+1]*(1-Phi.mat[j,j]^2)
}
round(Phi.mat[1:8,1:8],10)

+      [,1]      [,2] [,3] [,4] [,5] [,6] [,7] [,8]
+ [1,] -0.8798856 0.0000000 0.00 0 0 0 0 0
+ [2,] -1.5155993 -0.7224959 0.00 0 0 0 0 0
+ [3,] -1.4000000 -0.4800000 0.16 0 0 0 0 0
+ [4,] -1.4000000 -0.4800000 0.16 0 0 0 0 0
+ [5,] -1.4000000 -0.4800000 0.16 0 0 0 0 0
+ [6,] -1.4000000 -0.4800000 0.16 0 0 0 0 0
+ [7,] -1.4000000 -0.4800000 0.16 0 0 0 0 0
+ [8,] -1.4000000 -0.4800000 0.16 0 0 0 0 0

diag(Phi.mat)[1:10] #pacfs

+ [1] -8.798856e-01 -7.224959e-01 1.600000e-01 1.776357e-15 4.440892e-16
+ [6] -8.881784e-16 -8.881784e-16 -8.881784e-16 -4.440892e-16 0.000000e+00

ARMAacf(ar=Re(-ar.coefs[2:4]), lag.max = 10, pacf=TRUE) #Check

+ [1] -8.798856e-01 -7.224959e-01 1.600000e-01 1.847383e-15 1.696718e-17
+ [6] -6.351014e-16 -1.863144e-16 -8.370233e-16 -1.126903e-15 -2.665510e-16

v[1:20]

+ [1] 9.508426 2.147015 1.026273 1.000000 1.000000 1.000000 1.000000 1.000000
+ [9] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
+ [17] 1.000000 1.000000 1.000000 1.000000

```

The matrix `Phi.mat` contains the results of the Levinson-Durbin algorithm, and its diagonal elements are the PACF values match those computed by the `ARMAacf` function. Note that the PACF values are

$$\alpha_X(h) = \begin{cases} 1 & h = 0 \\ -0.8799 & h = 1 \\ -0.7225 & h = 2 \\ 0.16 & h = 3 \\ 0 & h \geq 4 \end{cases}$$

Also, note that the `Phi.mat` matrix has identical rows after the third. This is because for an AR(3) model, the optimal prediction is always the conditional mean. Note also that the prediction variance converges to a constant.

The Innovations Algorithm: Let $\{X_t\}$ be zero-mean process with $\mathbb{E}[X_t^2] < \infty$ and let $\gamma_X(t, s) = \mathbb{E}[X_t X_s]$. Let

$$\hat{X}_n = \begin{cases} 0 & \text{if } n = 1 \\ \mathbb{E}[X_n | \mathbf{X}_{1:(n-1)}] & \text{if } n \geq 2 \end{cases}$$

Let $U_n = X_n - \hat{X}_n$ be the error in prediction and let $\mathbf{U}_{1:n}$ be the vector of prediction errors (computed in a one-step ahead fashion). We assume that $\mathbb{E}[X_n | \mathbf{X}_{1:(n-1)}]$ is linear in $X_1, \dots, X_{n-1}$, and write $\mathbf{U}_{1:n} = \mathbf{A}_n \mathbf{X}_{1:n}$ where $\mathbf{A}_n$ is given by

$$\mathbf{A}_n = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ a_{1,1} & 1 & 0 & 0 & 0 \\ a_{2,2} & a_{2,1} & 1 & 0 & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1,n-1} & \cdots & \cdots & a_{n-1,1} & 1 \end{bmatrix}$$

as

$$\begin{aligned} U_1 &= X_1 \\ U_2 &= X_2 - \hat{X}_2 = X_2 + a_{1,1}X_1 \\ U_3 &= X_3 - \hat{X}_3 = X_3 + a_{2,2}X_2 + a_{2,1}X_1 \\ &\vdots \end{aligned}$$

Let $\mathbf{C}_n = \mathbf{A}_n^{-1}$, again a lower triangular matrix, where

$$\mathbf{C}_n = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \theta_{1,1} & 1 & 0 & 0 & 0 \\ \theta_{2,2} & \theta_{2,1} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{n-1,n-1} & \cdots & \cdots & \theta_{n-1,1} & 1 \end{bmatrix}$$

Then $\mathbf{X}_{1:n} = \mathbf{C}_n \mathbf{U}_{1:n} = \mathbf{C}_n (\mathbf{X}_{1:n} - \hat{\mathbf{X}}_{1:n})$, and

$$\hat{\mathbf{X}}_{1:n} = \mathbf{X}_{1:n} - \mathbf{U}_{1:n} = \mathbf{C}_n \mathbf{U}_{1:n} - \mathbf{U}_{1:n} = (\mathbf{C}_n - \mathbf{I}_n) \mathbf{U}_{1:n} = (\mathbf{C}_n - \mathbf{I}_n) (\mathbf{X}_{1:n} - \hat{\mathbf{X}}_{1:n}) = \boldsymbol{\Theta}_n (\mathbf{X}_{1:n} - \hat{\mathbf{X}}_{1:n})$$

where

$$\boldsymbol{\Theta}_n = \mathbf{C}_n - \mathbf{I}_n = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \theta_{1,1} & 0 & 0 & 0 & 0 \\ \theta_{2,2} & \theta_{2,1} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{n-1,n-1} & \cdots & \cdots & \theta_{n-1,1} & 0 \end{bmatrix}$$

and so

$$\hat{X}_{n+1} = \begin{cases} 0 & \text{if } n = 1 \\ \sum_{i=1}^n \theta_{n,i} (X_{n+1-i} - \hat{X}_{n+1-i}) & \text{if } n \geq 2 \end{cases}$$

i.e. $\hat{X}_{n+1}$ is a linear combination of $X_1 - \hat{X}_1, X_2 - \hat{X}_2, \dots, X_n - \hat{X}_n$, the one-step ahead prediction errors which are *uncorrelated*.

For the general (non-stationary) case:

- Initialize

$$v_0 = \tilde{\gamma}_X(1, 1) = \mathbb{E}[X_1^2]$$

- Recursion at step $i = 1, 2, 3, \dots$:

$$\theta_{i,i-k} = \frac{1}{v_k} \left[\tilde{\gamma}_X(i+1, k+1) - \sum_{j=0}^{k-1} \theta_{k,k-j} \theta_{i,i-j} v_j \right]$$

for $0 \leq k < i$

$$v_i = \tilde{\gamma}_X(i+1, i+1) - \sum_{j=0}^{i-1} \theta_{i,i-j}^2 v_j$$

Compute: $v_0, \theta_{1,1}, v_1, \theta_{2,2}, \theta_{2,1}, v_2, \theta_{3,3}, \theta_{3,2}, \theta_{3,1}, v_3, \dots$

For the stationary case:

- Initialize

$$v_0 = \gamma_X(0) = \mathbb{E}[X_1^2]$$

- Recursion at step $i = 1, 2, 3, \dots$:

$$\theta_{i,i-k} = \frac{1}{v_k} \left[\gamma_X(k-i) - \sum_{j=0}^{k-1} \theta_{k,k-j} \theta_{i,i-j} v_j \right]$$

for $0 \leq k < i$

$$v_i = \gamma_X(0) - \sum_{j=0}^{i-1} \theta_{i,i-j}^2 v_j$$

Compute: $v_0, \theta_{1,1}, v_1, \theta_{2,2}, \theta_{2,1}, v_2, \theta_{3,3}, \theta_{3,2}, \theta_{3,1}, v_3, \dots$

```
Gamma.X<-Variance*ARMAacf(ar=Re(-ar.coefs[2:4]), lag.max = n+1)
v<-rep(0,n+1)
v[0+1]<-Gamma.X[0+1] #Index 0
Theta.mat<-matrix(0,n+1,n+1)
Theta.mat[1,1]<-Gamma.X[1+1]/v[0+1] #Index i=1
v[1+1]<-Gamma.X[0+1]-Theta.mat[1,1]^2*v[0+1]
for(i in 2:n){ #Index i=2,3,...
  for(k in 0:(i-1)){
    jvals<-0:(k-1)
    sumth<-sum(Theta.mat[k,k-jvals]*Theta.mat[i,i-jvals]*v[jvals+1])
    Theta.mat[i,i-k] = (Gamma.X[(i-k)+1] - sumth)/v[k+1]
  }
  jvals<-0:(i-1)
  v[i+1]<-(Gamma.X[0+1] - sum(Theta.mat[i,i-jvals]*Theta.mat[i,i-jvals]*v[jvals+1]))
}
v[1:20]

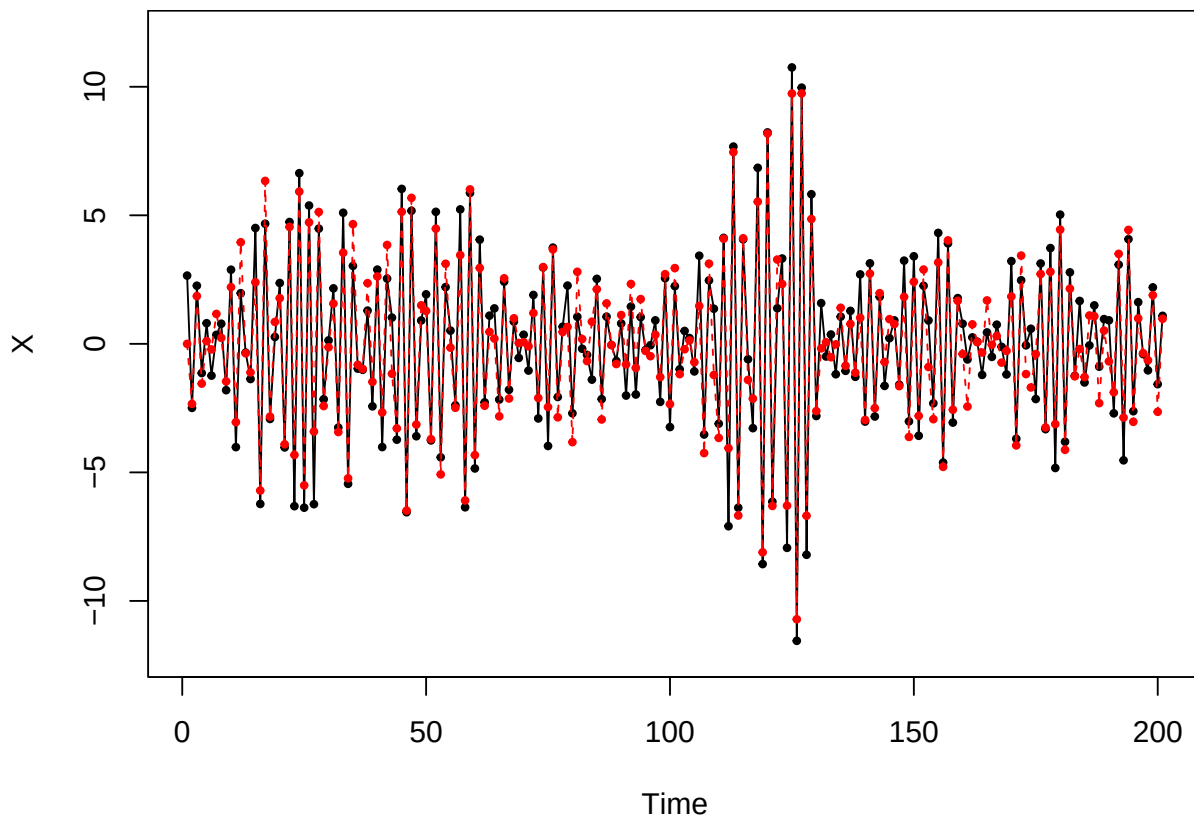
+ [1] 9.508426 2.147015 1.026273 1.000000 1.000000 1.000000 1.000000 1.000000
+ [9] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
+ [17] 1.000000 1.000000 1.000000 1.000000

Theta.mat[1:5,1:5]

+           [,1]      [,2]      [,3]      [,4]      [,5]
+ [1,] -0.8798856 0.0000000 0.0000000 0.0000000 0.0000000
+ [2,] -1.5155993 0.6110582 0.0000000 0.0000000 0.0000000
+ [3,] -1.4000000 1.6418391 -0.2731363 0.0000000 0.0000000
+ [4,] -1.4000000 1.4800000 -1.4110870 -0.05169876 0.0000000
+ [5,] -1.4000000 1.4800000 -1.2400000 0.94494319 0.301253
```

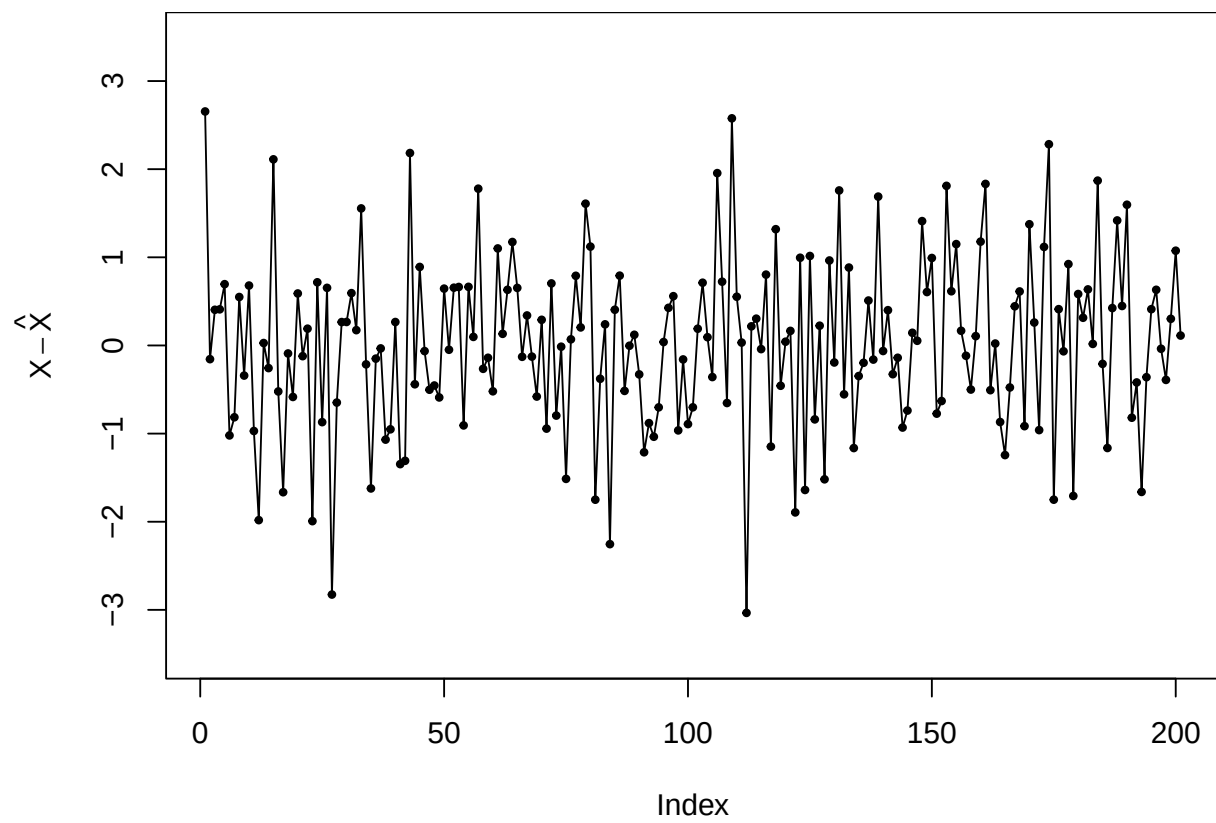
For a real series, we can see how the forecast behaves:

```
set.seed(26)
n<-200
X<-arima.sim(n=n+1,model=list(ar=Re(-ar.coefs[2:4])))
Xhat<-rep(0,n+1)
for(ni in 1:n){
  Xhat[ni+1]<-Theta.mat[ni,1:ni] %*% rev(X[1:ni] - Xhat[1:ni])
}
par(mar=c(4,6,0,1))
plot(X,type='l',ylim=range(-12,12))
points(X,pch=19,cex=0.5)
lines(Xhat,type='l',lty=2,col='red')
points(Xhat,pch=19,cex=0.5,col='red')
```



The one-step ahead forecasts seem largely to be accurate.

```
par(mar=c(4,6,0,1))
plot(as.numeric(X-Xhat),type='l',ylim=range(-3.5,3.5),ylab=expression(X-hat(X)))
points(as.numeric(X-Xhat),pch=19,cex=0.5)
```



```
acf(as.numeric(X-Xhat),lag.max=20,ylim=range(-1,1))
```

