# Lecture 7 The Levinson Algorithm

Let $\{X_k\}_{k=-\infty}^{\infty}$ be a *wide sense stationary* (*wss*) zero-mean discrete-time random process. Let $\{E(X_k X_{k+m}) = R_X(m)\}_{m=0}^{\infty}$ denote the *autocorrelation function* for this process. We begin this set of notes with the following problem.

**The Linear Prediction Problem**: In words, the problem here is to predict $X_k$ from $\{X_{k-j}\}_{j=1}^{p}$ using the following linear prediction model:

$$\widehat{X}_k^{(p)} = -\sum_{j=1}^{p} a_{p,j} X_{k-j} . \tag{1}$$

The integer, *p*, is the number of *lags* (relative to *k*) that are to be used to predict $X_k$. Consequently, (1) is called a $p^{th}$ *order linear prediction model.* The corresponding $p^{th}$ *order prediction error* is clearly,

$$X_k - \widehat{X}_k^{(p)} \overset{\Delta}{=} V_k^{(p)} . \tag{2}$$

Since Let $\{X_k\}_{k=-\infty}^{\infty}$ is a *wss* zero-mean random process, so is the error random process $\{V_k^{(p)}\}_{k=-\infty}^{\infty}$. Hence, the *mean-squared error* (*mse*) is $E[(V_k^{(p)})^2] \overset{\Delta}{=} \sigma_p^2$. We desire to find the prediction coefficients $\{a_{p,j}\}_{j=1}^{p}$ that will minimize this *mse.*

***Fact* 1:** The prediction coefficients $\{a_{p,j}\}_{j=1}^{p}$ that minimize $\sigma_p^2$, are those that satisfy the following *orthogonality conditions*:

$$E(V_k^{(p)} X_{k-m}) = 0 \quad \text{for } m = 1, 2, \ldots, p . \tag{3}$$

Clearly, (3) includes *p* linear equations in the *p* unknowns $\{a_{p,j}\}_{j=1}^{p}$. To arrive at the explicit set of equations, use (1) and (2) to express (3) as:

$$E(V_k^{(p)} X_{k-m}) = E[(X_k - \widehat{X}_k^{(p)}) X_{k-m}] = R_X(m) + \sum_{j=1}^{p} a_{p,j} R_X(j-m) = 0 \quad \text{for } m = 1, 2, \ldots, p . \tag{4}$$

The collection of equations in (4) can be written in the following matrix form:

$$\begin{bmatrix} R_X(0) & R_X(1) & \cdots & R_X(p-1) \\ R_X(1) & R_X(0) & \cdots & R_X(p-2) \\ \vdots & \vdots & \vdots & \vdots \\ R_X(p-1) & R_X(p-2) & \cdots & R_X(0) \end{bmatrix} \begin{bmatrix} a_{p,1} \\ a_{p,2} \\ \vdots \\ a_{p,p} \end{bmatrix} = -\begin{bmatrix} R_X(1) \\ R_X(2) \\ \vdots \\ R_X(p) \end{bmatrix} . \tag{5a}$$

We will express (5a) in the following concise notation:

$$\Re_{p-1}\,\alpha_p \;=\; \rho_p\ . \tag{5b}$$

Hence, the vector of prediction coefficients $\alpha_p = [a_{p,1} \quad a_{p,2} \quad \cdots a_{p,p}]^{tr}$ corresponding to the *minimum mean-squared error* (*mmse*) $p^{th}$ *order* linear prediction model of the form (1) is:

$$\alpha_p \;=\; \Re_{p-1}^{-1}\,\rho_p\ . \tag{6a}$$

The corresponding *mse* is:

$$\sigma_p^2 = E[(V_k^{(p)})^2] = E[(V_k^{(p)}\,(X_k - \widehat{X}_k)] = E(V_k^{(p)}\,X_k) - E(V_k^{(p)}\widehat{X}_k) = E(V_k^{(p)}\,X_k) = R_X(0) + \sum_{j=1}^{p} a_{p,j} R_X(j)\,.$$

Notice that the term $E(V_k^{(p)}\widehat{X}_k)$ vanished. This is because the prediction error $V_k^{(p)}$ is uncorrelated with every member of $\{X_{k-j}\}_{j=1}^{p}$, and so it is uncorrelated with $\widehat{X}_k$. The above equation can be written in the more concise form:

$$\sigma_p^2 \;=\; R_X(0) \;+\; \alpha_p^{tr}\,\rho_p\,. \tag{6b}$$

Equations (6) represent the solution to the $p^{th}$ *order* linear prediction problem. The crucial issue that was not addressed is how to select the most appropriate order, *p*, for a given collection autocorrelations $\{R_X(m)\}_{m=0}^{M}$. Now if these autocorrelations are known to be *exact*, then this is not a problem, since we would simply choose $p = M$, as that will result in the smallest possible *mse*. If the *mmse* is actually achieved for $p < M$, then the $M^{th}$ *order* coefficient vector is $\alpha_M^{tr} = [\alpha_p^{tr}\ 0,\ldots,0]$. In this idealistic case the random process $\{X_k\}_{k=-\infty}^{\infty}$ is called a $p^{th}$ *order autoregressive* [AR(p)] *process.* We state this as

***Definition* 1.** If a *wss* zero-mean process $\{X_k\}_{k=-\infty}^{\infty}$ can be expressed as $X_k \;=\; -\sum_{j=1}^{p} a_{p,j} X_{k-j} + V_k$, where $\{V_k\}_{k=-\infty}^{\infty}$ is a *white noise* process, then it is called a $p^{th}$ *order autoregressive* [AR(p)] *process.*

Hence, in the case of an AR(*p*) process, not only does the orthogonality condition hold for $m = 1, 2, \ldots, p$, it hold for *all* $m>0$.

If, indeed, this minimum possible *mse* can be achieved for a model order $p << M$, then since $\alpha_M^{tr} = [\alpha_p^{tr}\ 0,\ldots,0]$, the $M^{th}$ order model collapses to a $p^{th}$ order AR(*p*) model. Few, if any real-world random processes are *truly* AR(*p*) in nature; albeit many can be well-modeled by the same. Recognition of the following fact is central to the use of AR(*p*) models.

**Fact 2.** The solution (6a) of the equation (5a) guarantees that (2) will be uncorrelated with the collection $\{X_{k-j}\}_{j=1}^{p}$. However, this does not mean that (2) is a *white noise* process. It will be a white noise process if and only if (1) is an AR(*p*) process. When this is not the case, then (2) will be a *colored noise* process; that is, it will retain some of the correlation structure related to (1). To capture this structure would require a higher order model.

Another problem is that of not having *exact* knowledge of $\{R_X(m)\}_{m=0}^{M}$. Typically, we have data-based *estimates* $\{\hat{R}_X(m)\}_{m=0}^{M}$. And so, whereas in the ideal case the last *M-p* elements of $\alpha_M^{tr} = [\alpha_p^{tr}\ 0,\ldots,0]$ would be exactly zero, in this more common and realistic case they will not. The most common estimator of $R_X(m)$ is the following *lagged-product* estimator:

$$\hat{R}_X(m) \quad = \quad \frac{1}{n}\sum_{k=1}^{n-m} X_k X_{k+m}\ . \tag{7}$$

Notice that for $m = 0$, (7) is the average of *n* products. Hence, if the observation length, *n*, is large, (7) will be a good estimator of $R_X(0)$. At the other extreme, suppose that $m = n-1$, which is the largest value of *m* that can be used in (7). In this case, $\hat{R}_X(n-1) \quad = \quad \frac{1}{n} X_1 X_n$. This is not an average at all. It is (1/*n*) times a single product. As a result, $\hat{R}_X(n-1)$ will be a *very* poor estimator of $R_X(n-1)$.

We will quantify the quality of the estimator (7) in due course. For now, it is enough to recognize that if the estimators (7) for $m = 1, 2, \ldots, p$ are used in (5a), then as *p* increases for a given data length, *n*, (a) will include more and more poor estimators of the higher autocorrelation lags. Consequently, the estimator (6a) will become less trustworthy.

It is this trade-off between the desire for a high model order that can better capture the structure of the process, and the increasing uncertainty of the estimator (7) at higher lags that has led researchers to propose a wide variety of model order identification schemes. All of these schemes represent an attempt to somehow optimize this trade-off. All of them strive to identify that *single* 'best' model order, *p*. And so, to this end, (6) will be computed for a variety of increasing model orders.

**The Levinson Algorithm**

Notice that the computations, (6), involve taking the inverse of the $p \times p$ matrix $\Re_{p-1}$ in (5b), as defined by (5a). Before the advent of high speed computers, computing such an inverse became exponentially more intensive as the order *p* increased. Today, even for *p* on the order of 100, such an inverse can be computed in practically no time. The *Levinson* algorithm was developed in the mid-1960's as an alternative to having to perform the matrix inversion. Even though current computing power has lessened its value, we include it here for two reasons. First, it can be implemented in a *digital signal processing* (*DSP*) chip far more cheaply that the matrix inversion method. Second, we will see that by progressing through the order sequence $p = 1,2,3,\ldots$, not only do we arrive at a *family* of AR models that can be used for cross-validation purposes, but we also arrive at a family of related *minimum variance* (*MV*) models that include information about the process not so easily gleaned from the *AR* models.

To arrive at the *Levinson algorithm*, we begin with the *p* orthogonality conditions (4), which we give here for convenience:

$$R_X(m) + \sum_{j=1}^{p} a_{p,j} R_X(j-m) \quad = \quad 0 \quad \text{for } m = 1, 2, \ldots, p\,. \tag{8a}$$

and with the equation following (6a), which led to (6b):

$$\sigma_p^2 = R_X(0) + \sum_{j=1}^{p} a_{p,j} R_X(j)\,. \tag{8b}$$

Equations (8) can be written as:

$$\begin{bmatrix} R_X(0) & R_X(1) & \cdots & R_X(p-1) & R_X(p) \\ R_X(1) & R_X(0) & \cdots & R_X(p-2) & R_X(p-1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ R_X(p-1) & R_X(p-2) & \cdots & R_X(0) & R_X(1) \\ R_X(p) & R_X(p-1) & \cdots & R_X(1) & R_X(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_{p,1} \\ \vdots \\ a_{p,p-1} \\ a_{p,p} \end{bmatrix} = \begin{bmatrix} \sigma_p^2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{9a}$$

We will now define $a_{p,0} = 1$, and subsequently, *re-define* $\alpha_p = [a_{p,0}, a_{p,1}, \cdots, a_{p,p}]^{tr}$, so that (9a) is:

$$\Re_p\, \alpha_p \quad = \quad \begin{bmatrix} \sigma_p^2 \\ \vec{0} \end{bmatrix}. \tag{9b}$$

The matrix $\Re_p$ is not only symmetric, but the $k^{th}$ diagonal contains the single element $R_X(k)$. Such a matrix is called a *Toeplitz* matrix, and it has the following property:

$$\Re_p\, \breve{\alpha}_p \quad = \quad \begin{bmatrix} \vec{0} \\ \sigma_p^2 \end{bmatrix} \qquad \text{where} \qquad \breve{\alpha}_p = [a_{p,p} \quad a_{p,p-1} \quad \cdots \quad a_{p,1}\, a_{p,0}]^{tr}. \tag{10}$$

Now, we also have

$$\begin{bmatrix} R_X(0) & R_X(1) & \cdots & R_X(p-1) & R_X(p) \\ R_X(1) & R_X(0) & \cdots & R_X(p-2) & R_X(p-1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ R_X(p-1) & R_X(p-2) & \cdots & R_X(0) & R_X(1) \\ R_X(p) & R_X(p-1) & \cdots & R_X(1) & R_X(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_{p-1,1} \\ \vdots \\ a_{p-1,p-1} \\ 0 \end{bmatrix} = \begin{bmatrix} \sigma_{p-1}^2 \\ 0 \\ 0 \\ 0 \\ \Delta_{p-1} \end{bmatrix} \tag{11a}$$

where $\Delta_{p-1} = [\alpha_{p-1}^{tr}\ 0][R_X(p)\ R_X(p-1)\cdots R_X(0)]^{tr} \quad = \quad [\alpha_{p-1}^{tr}\ 0]\breve{\rho}_p$. (11b)

Hence, in compact form, (11) becomes:

$$\Re_p \begin{bmatrix} \alpha_{p-1} \\ 0 \end{bmatrix} \quad = \quad \begin{bmatrix} \sigma_{p-1}^2 \\ \vec{0} \\ \Delta_{p-1} \end{bmatrix}. \tag{12a}$$

Similar to (10), we have from (12a):

$$\Re_p \begin{bmatrix} 0 \\ \breve{\alpha}_{p-1} \end{bmatrix} \quad = \quad \begin{bmatrix} \Delta_{p-1} \\ \vec{0} \\ \sigma_{p-1}^2 \end{bmatrix}. \tag{12b}$$

***Claim*:** We can express $\alpha_p \quad = \quad \begin{bmatrix} \alpha_{p-1} \\ 0 \end{bmatrix} \quad - \quad \gamma \begin{bmatrix} 0 \\ \breve{\alpha}_{p-1} \end{bmatrix}$ for some value of $\gamma$.

To prove this claim, we will proceed to assume it is true, and find the appropriate value for $\gamma$.

$$\Re_p \alpha_p \quad = \quad \Re_p \left\{ \begin{bmatrix} \alpha_{p-1} \\ 0 \end{bmatrix} \quad - \quad \gamma \begin{bmatrix} 0 \\ \breve{\alpha}_{p-1} \end{bmatrix} \right\} = \begin{bmatrix} \sigma_{p-1}^2 - \gamma\Delta_{p-1} \\ \vec{0} \\ \Delta_{p-1} - \gamma\sigma_{p-1}^2 \end{bmatrix}. \tag{13}$$

If we compare it to (9b), we see that if we set

$$\gamma = \Delta_{p-1} / \sigma_{p-1}^2 \ ; \qquad \alpha_p \quad = \quad \begin{bmatrix} \alpha_{p-1} \\ 0 \end{bmatrix} \quad - \quad \gamma \begin{bmatrix} 0 \\ \breve{\alpha}_{p-1} \end{bmatrix}, \text{ and} \qquad \begin{bmatrix} \sigma_p^2 \\ \vec{0} \end{bmatrix} = \begin{bmatrix} \sigma_{p-1}^2 - \gamma\Delta_{p-1} \\ \vec{0} \end{bmatrix}$$

then we have *exactly* (9b). And so, the algorithm proceeds as follows:

*The Levinson Algorithm*:

*p=0*: $\alpha_0 = 1 \ ; \ \sigma_0^2 = R_X(0) \ ; \ \Delta_0 = R_X(1) \ ;$

*p=1* : $\gamma = \Delta_{p-1} / \sigma_{p-1}^2 = R_X(1) / R_X(0) \ ; \qquad \begin{bmatrix} 1 \\ a_{11} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \gamma \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -\gamma \end{bmatrix} \quad ; \quad \sigma_1^2 = R_X(0) - \gamma R_X(0) \ ;$

*p=2:* $\Delta_1 = [\alpha_1^{tr}\ 0][R_X(2)\ \ R_X(1)\ \ R_X(0)]^{tr} ; \ \gamma = \Delta_{p-1} / \sigma_{p-1}^2 \ ; \quad \alpha_p = \begin{bmatrix} \alpha_{p-1} \\ 0 \end{bmatrix} - \gamma \begin{bmatrix} 0 \\ \breve{\alpha}_{p-1} \end{bmatrix} \ ; \ \sigma_p^2 = \sigma_{p-1}^2 - \gamma\Delta_{p-1}$

*p=3:* $\Delta_2 = [\alpha_2^{tr}\ 0]\breve{\rho}_3 \ ; \qquad \gamma = \Delta_{p-1} / \sigma_{p-1}^2 \ ; \quad \alpha_p = \begin{bmatrix} \alpha_{p-1} \\ 0 \end{bmatrix} - \gamma \begin{bmatrix} 0 \\ \breve{\alpha}_{p-1} \end{bmatrix} \ ; \ \sigma_p^2 = \sigma_{p-1}^2 - \gamma\Delta_{p-1}$

The sequence of computations continues for as many models as are specified, up to order *n*-1.

### A *Matlab* Code for the Levinson Algorithm:

```
% The correlations Rx(k) for k = 0: maxorder must be resident as a column vector

E2=[];          %Array of model mse's
   Alpha=[];    %Array of model parameters. The kth column corresponds to {1 ak1 … akk}
   E2(1)=r(1); % This is actually Rx(0), but Matlab doesn't like the zero index.
   R=r(1:2);
   Alpha(1)=1.0;
   Aall=[Alpha; zeros(maxorder,1)];  % This array will have maxorder +1 rows of models.
   N=1;
   for n=1:maxorder
      R=r(1:n+1);
      rflip=flipud(R);
      Alpha=[Alpha; 0.0];
      del=rflip' * Alpha;
      Alpha=Alpha - (del/E2(N)) * flipud(Alpha);
      E2(N+1) = E2(N) - (del^2)/E2(N);
      N=n+1;
      Aall=[Aall,[Alpha; zeros(maxorder-n,1)]];
   end
```

*Example* 1. To test the above algorithm, we consider an AR(1) process, $X_k$ with $\alpha = 0.5$, and with $R_X(0)=1$. We define pwr = 0:5 and r = (0.5*ones(6,0)).^pwr. This gives the first 6 autocorrelation lags (0:5). The resulting array of model coefficients is:

Aall =

| | | | | | |
|---|---|---|---|---|---|
| 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 0 | -0.5000 | -0.5000 | -0.5000 | -0.5000 | -0.5000 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

The corresponding array of *mse's* is:

E2 =

| | | | | | |
|---|---|---|---|---|---|
| 1.0000 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 0.7500 |

As expected, all higher order models collapse to the correct AR(1) model. □

### Using Simulations to Determine the Minimum Data length, *n*, for Acceptable Model Estimation

The above example utilized theoretical correlation lags. consequently, all models collapsed to the correct one. Suppose that these lags were, instead, estimated via (7). The question addressed here is:

*How large should the data length, n, be, in order to correctly identify the model as an AR(1) model?*

As mentioned above, we will pursue a theoretical answer to this question; one that utilizes a fair bit of probability theory. However, in view of the level of computational power presently available, the student can answer this question by performing simulations. Let's begin by simulating the above AR(1) process for various values of *n*.

***Case* 1: *n*=100** The estimated autocorrelations from ar1sim.m (given below) are:

Trial>> Rhat'

ans = 0.9842 0.4572 0.2514 0.2282 0.2011 0.1703

```
% PROGRAM NAME: ar1sim.m
a=0.5; varu=1-a^2;
n = 100; ntot = n+500;
u=varu^0.5 *randn(ntot,1);
x=zeros(ntot,1); x(1)=0;
for k = 2:ntot
    x(k) = a*x(k-1) + u(k);
end
x=x(501:ntot);
Rhat = xcorr(x,5,'biased');
Rhat=Rhat(6:11)
```

Using these in the scar.m program gives:

Aall =

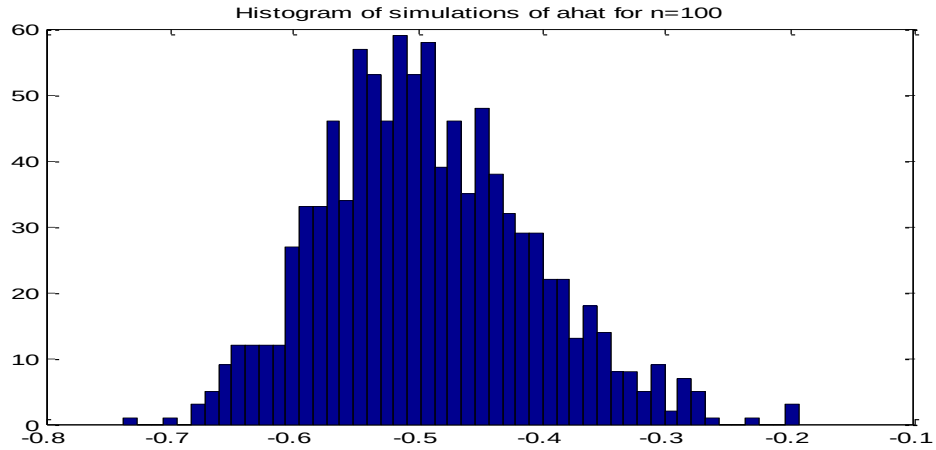
| | | | | | |
|---|---|---|---|---|---|
| 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 0 | -0.4646 | -0.4411 | -0.4350 | -0.4274 | -0.4247 |
| 0 | 0 | -0.0505 | 0.0036 | 0.0033 | 0.0075 |
| 0 | 0 | 0 | -0.1225 | -0.0957 | -0.0958 |
| 0 | 0 | 0 | 0 | -0.0616 | -0.0429 |
| 0 | 0 | 0 | 0 | 0 | -0.0436 |

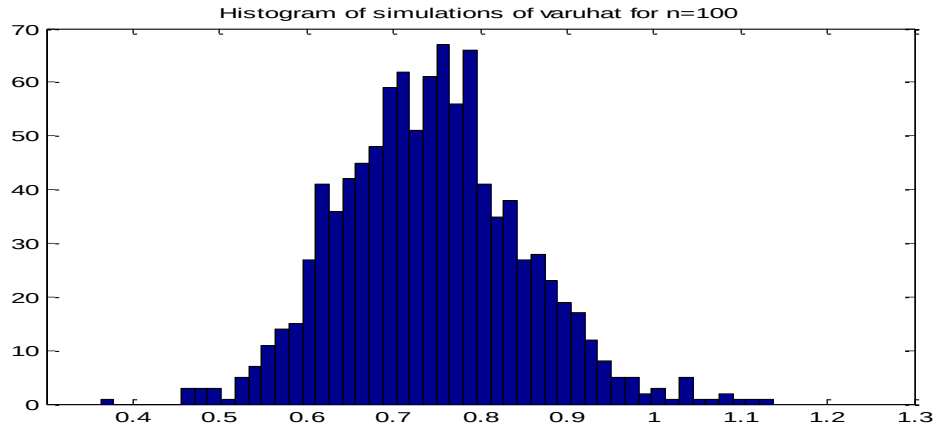
The array of corresponding *mse's* is:

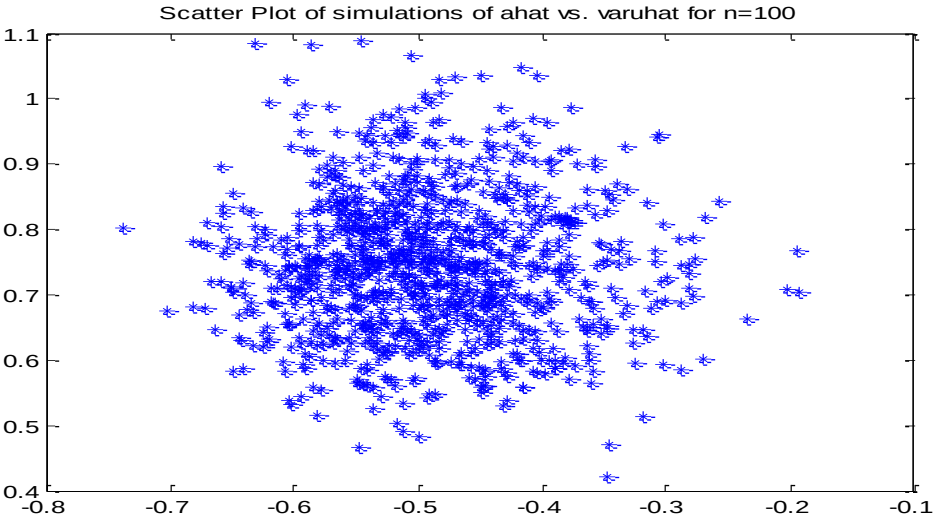
E2 = 0.9842 0.7718 0.7698 0.7583 0.7554 0.7540

We see that the *mse* decreases monotonically, but that the decrease is minimal beyond order 1. Hence, any model identification scheme would identify 1 as the best order. If we accept that for *n*=100, the order 1 would be identified, we can then proceed to investigate the *uncertainty* associated with the estimators of the AR(1) model parameters $\alpha$ and $\sigma_U^2$. These parameters depend only on $R_x(0)$ and $R_x(1)$. Specifically, $a = -R_x(1)/R_x(0)$, and $\sigma_U^2 = R_X(0) + aR_x(1)$. In this particularly simple setting, it is easier to forego the above codes and write a very simple direct one instead. To this end, consider the following code:

```
% PROGRAM NAME: ar1pdf.m
a=0.5; varu=1-a^2;
n = 100; ntot = n+500;
nsim = 1000;
u=varu^0.5 *randn(ntot,nsim);
x=zeros(ntot,nsim); x(1,:)=zeros(1,nsim);
for k = 2:ntot
    x(k,:) = a*x(k-1,:) + u(k,:);
end
x=x(501:ntot,:);
R0=mean(x.*x);
x0=x(1:n-1,:);
x1=x(2:n,:);
R1=mean(x0.*x1);
ahat = -R1./R0;
varuhat = R0 + ahat.*R1;
figure(1)
hist(ahat,50)
title('Histogram of simulations of ahat for n=100')
pause
figure(2)
hist(varuhat,50)
title('Histogram of simulations of varuhat for n=100')
pause
figure(3)
plot(ahat,varuhat,'*')
title('Scatter Plot of simulations of ahat vs. varuhat for n=100')
```


Histogram of simulations of ahat for n=100
60
50
40
30
20
10
0
-0.8
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1


Histogram of simulations of varuhat for n=100
70
60
50
40
30
20
10
0
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3


Scatter Plot of simulations of ahat vs. varuhat for n=100
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
-0.8
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1

**Estimating the Autocorrelation Function from the AR(p) model parameters**

After obtaining the AR($p$) model parameter estimates from the use of the Levinson algorithm in relation to the lagged-product autocorrelation estimates (7) up to order $p$, it is a simple matter to use the model to recursively estimate as many higher order lags as is desired. Specifically,

$$\widehat{R}_X(m) = -\sum_{j=1}^{p} a_{p,j} R_X(m-j) \quad \text{for } m > p\,. \tag{14}$$

The lagged-product autocorrelation estimator (7) is limited to $m < n$. Higher lags are implicitly presumed to be zero. This truncation of the higher lags is known as *windowing*, and has the effect of limiting the *spectral resolution* (to be discussed presently) to the order of the inverse of the window width. In contrast, (14) has no such truncation, as an arbitrary number of larger lags can be recursively computed. For this reason, AR models are also know as *high resolution spectral estimators.*