Linear predictive coding

This method combines linear processing with scalar quantization.

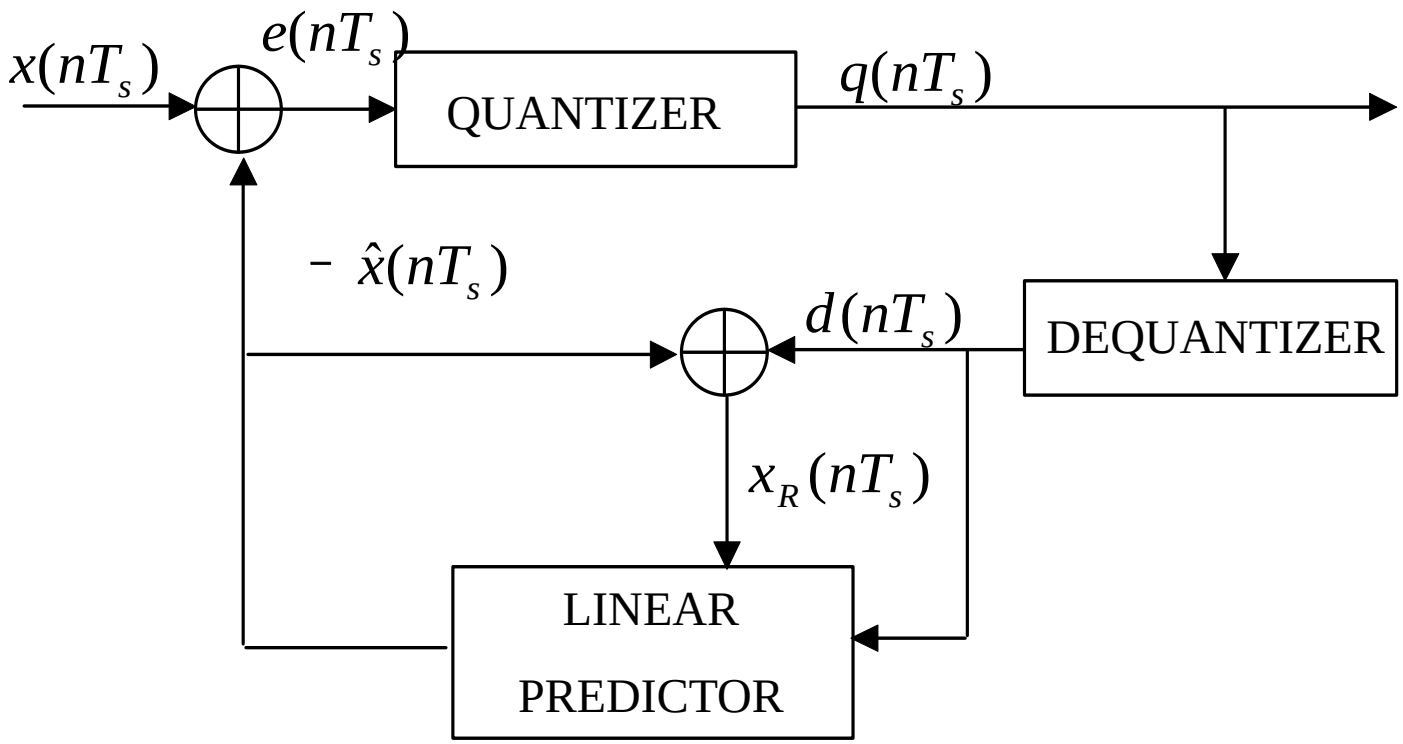
The main idea of the method is to predict the value of the current sample by a linear combination of previous already reconstructed samples and then to quantize the difference between the actual value and the predicted value.

Linear prediction coefficients are weighting coefficients used in linear combination.

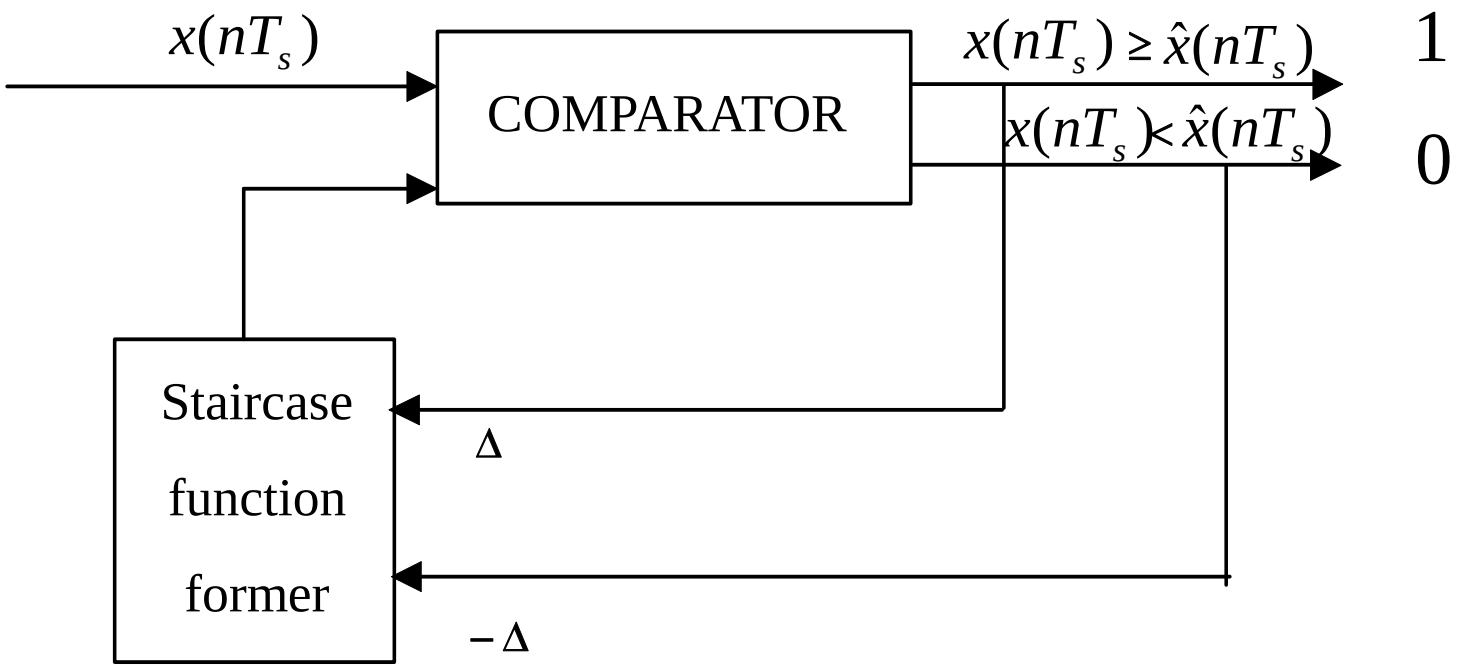
A simple predictive quantizer or differential pulse-coded modulator is shown in Fig. 5.1.

If the predictor is simply the last sample and the quantizer has only one bit, the system becomes a delta-modulator. It is shown in Fig. 5.2.

Differential pulse-coded modulator



Delta-modulator



Linear predictive coding

The main feature of the quantizers shown in Figs 5.1, 5.2 is that they exploit not all advantages of predictive coding.

- Prediction coefficients used in these schemes are not optimal
- Prediction is based on past reconstructed samples and not true samples
- Usually coefficients of prediction are chosen by using some empirical rules and are not transmitted

For example quantizer in Fig.5.1 instead of actual value of error e uses reconstructed values d and instead of true sample values x their estimates x_R obtained via d.

Linear predictive coding

The most advanced quantizer of linear predictive type represents a basis of the so-called Code Excited Linear Predictive (CELP) coder.

- It uses the optimal set of coefficients or in other words linear prediction coefficients of this quantizer are determined by minimizing the MSE between the current sample and its predicted value.
- It is based on the original past samples
- Using the true samples for prediction requires the "looking-ahead" procedure in the coder.
- The predictor coefficients are transmitted

Linear predictive coding

Assume that prediction coefficients are optimized for each frame and that the original past samples are used for prediction. Let

$x(T_s), x(2T_s), \ldots$ be a sequence of samples at the quantizer input. Then each sample $x(nT_s)$ is predicted by the previous samples according to the formula

$$\hat{x}(nT_s) = \sum_{k=1}^{m} a_k x(nT_s - kT_s), \quad (5.1)$$

where $\hat{x}(nT_s)$ is the predicted value, a_k are prediction coefficients, m denotes the order of prediction. The prediction error is

$$e(nT_s) = x(nT_s) - \hat{x}(nT_s).$$

Linear predictive coding

Prediction coefficients are determined by minimizing the sum of squared errors over a given interval (frame)

$$E = \sum_{n=n_0}^{n_1} e^2(nT_s) \qquad (5.2)$$

Inserting (5.1) into (5.2) we obtain

$$E = \sum_{n=n_0}^{n_1} (x(nT_s) - a_1 x(nT_s - T_s) - \ldots - a_m x(nT_s - mT_s))^2 =$$

$$= \sum_{n=n_0}^{n_1} \{x(nT_s)\}^2 - 2\sum_{j=1}^{m} a_j \sum_{n=n_0}^{n_1} x(nT_s)x(nT_s - jT_s) +$$

$$+ \sum_{j=1}^{m}\sum_{k=1}^{m} a_j a_k \sum_{n=n_0}^{n_1} x(nT_s - jT_s)x(nT_s - kT_s). \qquad (5.3)$$

Linear predictive coding

Differentiating (5.3) over a_k, $k = 1,2,...,m$ yields

$$\partial E / \partial a_k = \sum_{n=n_0}^{n_1} x(nT_s)x(nT_s - kT_s) - \sum_{j=1}^{m} a_j \sum_{n=n_0}^{n_1} x(nT_s - kT_s)x(nT_s - jT_s) = 0$$

Thus we obtain a system of m linear equations with m unknown quantities $a_1, a_2, ..., a_m$

$$\boxed{\sum_{j=1}^{m} a_j c_{jk} = c_{ok},} \quad k = 1,2,...,m \qquad (5.4)$$

where $c_{jk} = c_{kj} = \sum_{n=n_0}^{n_1} x(nT_s - jT_s)x(nT_s - kT_s).$ (5.5)

The system (5.4) is called the Yule-Walker equations.

Linear predictive coding

If $a_1, a_2, ..., a_m$ are solutions of (5.4) then we can find the minimal achievable prediction error. Insert (5.5) into (5.3). We obtain that

$$E = c_{00} - 2\sum_{k=1}^{m} a_k c_{0k} + \sum_{k=1}^{m} a_k \sum_{j=1}^{m} a_j c_{jk}. \qquad (5.6)$$

Using (5.3) we reduce (5.6) to

$$E = c_{00} - \sum_{k=1}^{m} a_k c_{0k}$$

Linear prediction as digital filtering

Eq. (5.1) describes the m th order predictor with transfer function equal to

$$P(z) = \frac{\hat{X}(z)}{X(z)} = \sum_{k=1}^{m} a_k z^{-k}.$$

z-transform for the prediction error is

$$E(z) = X(z) - \sum_{k=1}^{m} a_k X(z) z^{-k}.$$

The prediction error is an output signal of the discrete-time filter with transfer function

$$A(z) = \frac{E(z)}{X(z)} = 1 - \sum_{k=1}^{m} a_k z^{-k}.$$

The problem of finding the optimal set of prediction coefficients = problem of constructing m th order FIR filter.

Linear prediction as digital filtering

Another name of the linear prediction (5.1) is the autoregressive model (AM) of signal $x(nT_s)$. It is assumed that the signal $x(nT_s)$ can be obtained as the output of the autoregressive filter with transfer function

$$H(z) = \frac{1}{\left(1 - \sum_{k=1}^{m} a_k z^{-k}\right)},$$

that is can be obtained as the output of the filter which is inverse with respect to the prediction filter. This filter is a discrete-time IIR filter. The Y-W equations determine correspondence between parameters of AM and covariance function of the process.

Methods of finding coefficients

$$c_{ij}, i = 0,1,...,m,\ j = 1,2,...,m$$

In order to solve the Yule-Walker eq. (5.4) it is necessary first to evaluate values $c_{ij}, i = 0,1,...,m,\ j = 1,2,...,m$

There are two approaches to estimating these values:

The autocorrelation method and the covariance method.

The complexity of solving (5.4) is proportional to m^2

The complexity of solving (5.4) is proportional to m^3

Autocorrelation method

The values c_{ij} are computed as

$$c_{ij} = c_{ji} = \sum_{n=i_0}^{i_1} x(nT_s - iT_s)x(nT_s - jT_s). \qquad (5.7)$$

We set $i_0 = -\infty, i_1 = \infty$ and $x(nT_s) = 0$ if $n < 0, n \geq N$, where N is called the interval of analysis.

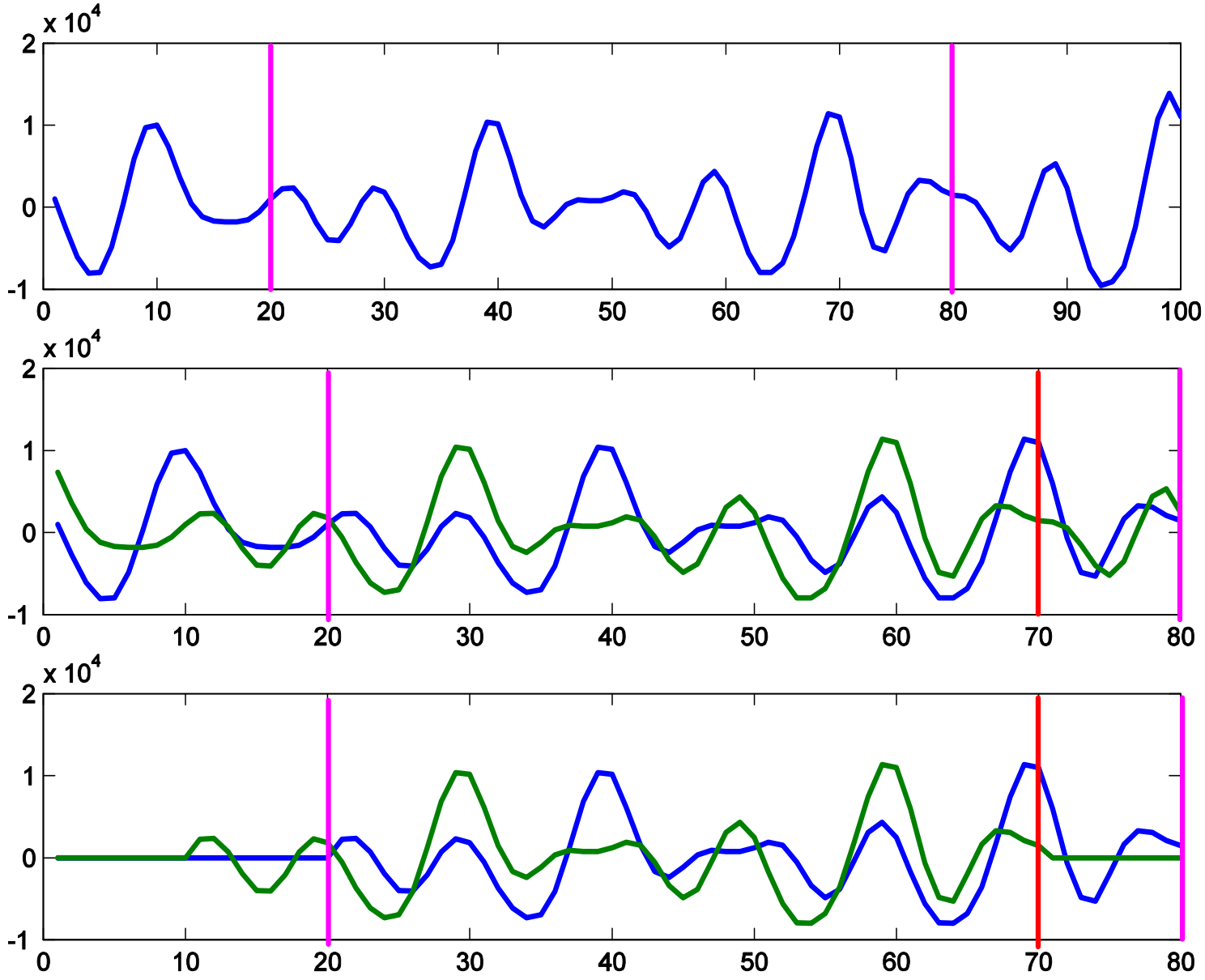
In this case we can simplify (5.7)

$$c_{ij} = \sum_{n=0}^{N-1} x(nT_s - iT_s)x(nT_s - jT_s) = \sum_{n=0}^{N-1-|i-j|} x(nT_s)x(nT_s + |i-j|T_s). \qquad (5.8)$$

Normalized by N they coincide with estimates of entries of covariance matrix for $x(nT_s)$

$$\hat{R}(|i-j|) = c_{ij} / N = 1/N \sum_{n=0}^{N-1-|i-j|} x(nT_s)x(nT_s + |i-j|T_s).$$

Autocorrelation method



Autocorrelation method

The Yule-Walker equations for autocorrelation method have the form

$$\sum_{i=1}^{m} a_i \hat{R}(|i-j|) = \hat{R}(j),\ j = 1,2,...,m. \qquad (5.9)$$

Eq.(5.9) can be given by matrix equation

$$\mathbf{a} \times \mathbf{R} = \mathbf{b},$$

where $\mathbf{a} = (a_1, a_2, ..., a_m)$, $\mathbf{b} = (\hat{R}(1), \hat{R}(2), ..., \hat{R}(m))$,

$$\mathbf{R} = \begin{bmatrix} \hat{R}(0) & \hat{R}(1)... & \hat{R}(m-1) \\ \hat{R}(1) & \hat{R}(0)... & \hat{R}(m-2) \\ ..... & & \\ \hat{R}(m-1) & \hat{R}(m-2)... & \hat{R}(0) \end{bmatrix}.$$

Autocorrelation method

It is said that (5.9) relates the parameters of the autoregressive model of m th order with the autocorrelation sequence.

Matrix $\mathbf{R}$ of the autocorrelation method has two important properties.

- It is symmetric, that is $\hat{R}(i, j) = \hat{R}(j, i)$
- It has the Toeplitz property, that is $\hat{R}(i, j) = \hat{R}(|i - j|)$.

The Toeplitz property of $\mathbf{R}$ makes it possible to reduce the computational complexity of solving (5.4). The fast Levinson-Durbin recursive algorithm requires only m^2 operations.

Covariance method

We choose $i_0 = 0$ and $i_1 = N-1$ and signal $x(nT_s)$ is not constrained in time. In this case we have

$$c_{ij} = \sum_{n=0}^{N-1} x(nT_s - iT_s)x(nT_s - jT_s). \qquad (5.10)$$

Set $k = n - i$ then (5.10) can be rewritten as

$$c_{ij} = \sum_{k=-i}^{N-i-1} x(kT_s)x(kT_s + (i-j)T_s), i = 1,...,m,\ j = 0,...m. \qquad (5.11)$$

(5.11) resembles (5.8) but it has other range of definition for k.

- It uses signal values out of range $0 \le k \le N-1$,
- The method leads to the cross-correlation function between two similar but not exactly the same finite fragments of $x(kT_s)$.

Covariance method

$$\hat{R}(i,j) = c_{ij} / N = 1/N \sum_{n=0}^{N-1} x((n-i)T_s)x((n-j)T_s).$$

The Yule-Walker equations for the covariation method are

$$\sum_{i=1}^{m} a_i \hat{R}(i,j) = \hat{R}(0,j),\ j = 1,2,...,m. \qquad (5.12)$$

Eq. (5.12) can be given by the matrix equation

$$\mathbf{a} \times \mathbf{P} = \mathbf{c},$$

where $\mathbf{a} = (a_1, a_2, ..., a_m), \mathbf{c} = (\hat{R}(0,1), \hat{R}(0,2), ..., \hat{R}(0,m)),$

$$\mathbf{P} = \begin{bmatrix} \hat{R}(1,1)\hat{R}(1,2)...\hat{R}(1,m) \\ \hat{R}(2,1)\hat{R}(2,2)...\hat{R}(2,m) \\ ..... \\ \hat{R}(m,1)\hat{R}(m,2)...\hat{R}(m,m) \end{bmatrix}.$$

Covariance method

Unlike the matrix $\mathbf{R}$ of autocorrelation method the matrix $\mathbf{P}$ is symmetric ($\hat{R}(i, j) = \hat{R}(j, i)$) but it is not Toeplitz .

Since computational complexity of solving an arbitrary system of linear equations of order m is equal to m^3 then in this case to solve (5.12) it is necessary m^3 operations.

Algorithms for the solution of the Yule-Walker equations

The computational complexity of solving the Yule-Walker equations depends on the method of evaluating values c_{ij}.

Assume that c_{ij} are found by the autocorrelation method.

In this case the Yule-Walker equations has the form (5.9) and the matrix $\mathbf{R}$ is symmetric and the Toeplitz matrix.

These properties make it possible to find the solution of (5.9) by fast methods requiring m^2 operations.

There are a few methods of this type: the Levinson-Durbin algorithm, the Euclidean algorithm and the Berlekamp-Massey algorithm.

The Levinson-Durbin algorithm

It was suggested by Levinson in 1948 and then was improved by Durbin in 1960. Notice that this algorithm works efficiently if matrix of coefficients $\mathbf{R}$ is simultaneously symmetric and Toeplitz. The Berlekamp-Massey and the Euclidean algorithms do not require the matrix of coefficients to be symmetric.

We sequentially solve equations (5.9) of order $l = 1,..., m.$

Let $\mathbf{a}^{(l)} = (a_1^{(l)}, a_2^{(l)},..., a_l^{(l)})$ denote the solution for the system of the l th order. Given $\mathbf{a}^{(l)}$ we find the solution for the $(l+1)$th order. At each step of the algorithm we evaluate the prediction error E_l of the l th order system.

The Levinson-Durbin algorithm

Initialization: $l = 0, E_0 = \hat{R}(0), a^{(0)} = 0.$

Recurrent procedure:

For $l = 1,...,m$ **compute**

$$a_l^{(l)} = (\hat{R}(l) - \sum_{i=1}^{l-1} a_i^{(l-1)} \hat{R}(l-i)) / E_{l-1},$$

$$a_j^{(l)} = a_j^{(l-1)} - a_l^{(l)} a_{l-j}^{(l-1)}, \quad 1 \le j \le l-1,$$

$$E_l = E_{l-1}(1-(a_l^{(l)})^2).$$

When $l = m$ we obtain the solution

$$\mathbf{a} = (a_1, a_2, ..., a_m) = \mathbf{a}^{(m)}, E = E_m.$$

Example

$m = 1 \qquad R(0)a_1^{(1)} = R(1),$

$$k_1 = -R(1)/R(0),$$

$$a_1^{(1)} = R(1)/R(0),$$

$$a_1^{(1)} = R(1)/R(0),$$

$$E_1 = R(0) - a_1^{(1)} R(1)$$

$$E_1 = (R^2(0) - R^{(2)}(1))/R(0).$$

$m = 2$

$$\begin{cases} R(0)a_1^{(2)} + R(1)a_2^{(2)} = R(1) \\ R(1)a_1^{(2)} + R(0)a_2^{(2)} = R(2), \end{cases}$$

$$a_1^{(2)} = \frac{R(1)}{R(0)}(1 - a_2^{(2)}) = a_1^{(1)} - a_2^{(2)} a_1^{(1)}$$

$$a_2^{(2)} = \frac{R(2) - R(1)a_1^{(1)}}{R(0) - a_1^{(1)} R(1)} = \frac{R(2) - R(1)a_1^{(1)}}{E_1}$$

Example

$$a_1^{(2)} = (R(1)R(0) - R(1)R(2))/(R^2(0) - R^2(1))$$

$$a_2^{(2)} = (R(0)R(2) - R(1)^2)/(R^2(0) - R^{(2)}(1)).$$

$$E_2 = E_1(1 - (a_2^{(2)})^2)$$

$$E_2 = R(0) - a_1^{(2)}R(1) - a_2^{(2)}R(2) = (R(0) - a_1^{(1)}R(1))(1 - (a_2^{(2)})^2) =$$

$$= E_1(1 - (a_2^{(2)})^2).$$