Polynomial Factorization Using Recursive Formulas for Complex Integrals

Zhou, Zhao-Ying; Åström, Karl Johan

1981

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-7223)/1-030/(1981)

POLYNOMIAL FACTORIZATION USING RECURSIVE FORMULAS FOR COMPLEX INTEGRALS

ZHOU ZHAO-YING
KARL JOHAN ÅSTRÖM

DEPARTMENT OF AUTOMATIC CONTROL
LUND INSTITUTE OF TECHNOLOGY
JUNE 1981

POLYNOMIAL FACTORIZATION

USING RECURSIVE FORMULAS FOR COMPLEX INTEGRALS

Zhou Zhao-ying

Karl Johan Åström

Department of Automatic Control

Lund Institute of Technology

April 1981

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AUTOMATIC CONTROL
Box 725
S 220 07 Lund 7 Sweden

Document name

Date of issue
June 1981

Document number
CODEN: LUTFD2/(TFRT-7223)/1-030/(1981)

Author(s)
Zhou Zhao-ying
Karl Johan Åström

Supervisor

Sponsoring organization

Title and subtitle
Polynomial factorization using recursive formulas for complex integrals

Abstract

The complex integrals

$$I(\ell) = \frac{1}{2\pi i} \oint \frac{\varrho\, A(z)A(z^{-1}) + B(z)B(z^{-1})}{C(z)C(z^{-1})} \frac{dz}{z^{\ell+1}}$$

can be used to solve the spectral factorization problem

$$P(z)P(z^{-1}) = \varrho\, A(z)A(Z^{-1}) + B(z)B(z^{-1})$$

which appears in LQG calculations. This method has high precision.

The report gives recursive formulas for evaluating $I(\ell)$ and two iterative algorithms for the spectral factorization. Two test programs are written in PASCAL. The calculation of $I(0)$ also gives a test for the stability of the polynomial $P(z)$ in each iteration.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
30

Recipient's notes

Security classification

Distribution: The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

CONTENT

1. INTRODUCTION

Consider a discrete time system governed by the model

$$A(z)\, y(t) = B(z)\, u(t) \tag{1.1}$$

Let the criterion be to minimize

$$J = \sum_{t=0}^{\infty} [\, y^2(t) + \varrho\, u^2(t) \,] \tag{1.2}$$

The solution of this optimization problem gives a closed loop system whose characteristic polynomial P(z) is given by

$$P(z)P(z^{-1}) = \varrho\, A(z)A(z^{-1}) + B(z)B(z^{-1}) \tag{1.3}$$

where P, A and B are polynomials with real coefficients.

$$P(z) = p_0 z^k + P_1 z^{k-1} + \ldots + p_k$$

$$A(z) = a_0 z^k + a_1 z^{k-1} + \ldots + a_k$$

$$B(z) = b_0 z^k + b_1 z^{k-1} + \ldots + b_k$$

and ϱ is a weighting factor. See Astrom (1979). For self-tuning regulators it is useful to have efficient algorithms to solve the design problem.

It can be shown that the right hand side of (1.3) can be factorized when $\varrho \geq 0$. Consider a stationary discrete stochastic process

$$y(t) = r\, A(z)\, w(t) + B(z)\, v(t) \tag{1.4}$$

where w(t) and v(t) are uncorrelated white noises and the coefficients a_i and b_i are real numbers. Consider also another stationary discrete time process with white noise input $\varepsilon(t)$

$$z(t) = P(z)\, \varepsilon(t) \tag{1.5}$$

where P(z) is a real coefficient polynomial. These two process are equivalent in the sense of that they have same spectral density function, i.e.

$$\Phi_y(\omega) = \Phi_z(\omega)$$

Let $\rho = r^2$, then (1.3) follows. (1.3) has several solutions because of

$$P(z)P(z^{-1}) = k_1 \prod_{i=1}^{k} (z - r_i)(z^{-1} - r_i)$$

$$= k_2 \prod_{i=1}^{k} (z - 1/r_i)(z^{-1} - 1/r_i) \qquad (1.6)$$

where k_1 and k_2 are constants. The resultant polynomial P(z) is the combination of all possible zeros, but there is only one combination with

$$| z_i | \leq 1 \quad \text{for } i=0,1,..k$$

which corresponds to a stable polynomial P(z).

The complex integral

$$I(\ell) = \frac{1}{2\pi i} \oint \frac{\rho\, A(z)A(z^{-1}) + B(z)B(z^{-1})}{C(z)C(z^{-1})} \frac{dz}{z^{\ell+1}} \qquad (1.7)$$

is related to the factorization problem. The efficient algorithm Astrom et al (1970) and Astrom (1970) for evaluating I(0) can be used to solve (1.7), and the calculation of I(0) can also give a test for the stability of the polynomial P(z), which is described in previous literatures.

This report is organized as follows. The iteration method and the relationship between $I(\ell)$ and coefficients of P(z) is described in section 2. The recursive formulas are given in section 3. A numerical example shows that the method based on the complex integral has high precision. Section 4 shows a reduced version of recursive formulae which reguires less computing times than the former. Two programs in PASCAL are given in appendices.

2. ITERATION METHOD AND RELATIONSHIPS

A method based on Newton iteration is suggested by Kucera et al (1976) and Kucera (1979) because it features monotone and essentially quadratic convergence. Define polynomials C(z) and X(z) with

$$\deg C_J(z) = \deg X_J(z) = \deg P_J(z)$$

where subscript J denotes the Jth iteration. The iteration is performed recursively by solving

$$C_J(z)X_J(z^{-1}) + X_J(z)C_J(z^{-1}) = \varrho\, A(z)A(z^{-1}) + B(z)B(z^{-1}) \qquad (2.1)$$

starting from a stable polymonial $C_0(z)$. The next polynomial in the iteration is then given by

$$C_{J+1}(z) = \frac{1}{2}\left[\, C_J(z) + X_J(z) \,\right] \qquad (2.2)$$

and the desired polynomial P(z) is obtained as

$$P(z) = \lim_{J\to\infty} C_J(z) \qquad (2.3)$$

In practice it only takes a few steps for C(z) to converge to the desired polynomial. The test

$$| c_{i,J+1} - c_{i,J} | < \text{eps} \qquad \text{for } i=0,1,..,n$$

where eps is a given small value, can be used to stop the recursion.

Equation (2.1) can be written as a set of linear equations

$$CX = 2\,(\varrho\, a + b) \qquad (2.4)$$

where C is a square matrix of dimension k given by

$$C = \begin{bmatrix}
2c_0 & 2c_1 & 2c_2 & \cdots & 2c_{k-2} & 2c_{k-1} & 2c_k \\
c_1 & c_0+c_2 & c_1+c_3 & \cdots & c_{k-3}+c_{k-1} & c_{k-2}+c_k & c_{k-1} \\
c_2 & c_3 & c_0+c_4 & \cdots & c_{k-4}+c_k & c_{k-1} & c_{k-2} \\
\cdots & & & & & & \cdots \\
c_{k-1} & c_k & 0 & \cdots & 0 & c_0 & c_1 \\
c_k & 0 & 0 & \cdots & 0 & 0 & c_0
\end{bmatrix}$$

and a and b are vectors of the same form

$$a = \begin{bmatrix} \sum_{i=0}^{k} a_i^2 \\ \sum_{i=0}^{k-1} a_i a_{i+1} \\ \sum_{i=0}^{k-2} a_i a_{i+2} \\ \cdots \\ \sum_{i=0}^{1} a_i a_{i+k-1} \\ a_0 a_k \end{bmatrix}$$

If C(z) is a stable polynomial, dividing both sides of (2.1) by $C(z)C(z^{-1})$ gives

$$\frac{C(z)X(z^{-1}) + X(z)C(z^{-1})}{C(z)C(z^{-1})} = \frac{2[\varrho\, A(z)A(z^{-1}) + B(z)B(z^{-1})]}{C(z)C(z^{-1})} = 2 \sum_{\ell=-\infty}^{\infty} I(\ell)\, z^{\ell} \qquad (2.5)$$

or

$$\frac{X(z)}{C(z)} = I(0) + 2 \sum_{\ell=-\infty}^{-1} I(\ell)\, z^{\ell} \qquad (2.6)$$

Applying long division to the left side of (2.6) we have

$$\frac{X(z)}{C(z)} = \frac{x_0}{c_0} + \frac{1}{c_0}\left(x_1 - \frac{c_1}{c_0} x_0 \right) z^{-1} + \cdots \qquad (2.7)$$

Equating coefficients of same powers of (2.6) and (2.7) gives

$$I(0) = x_0 / c_0$$

$$I(1) = (x_1 - x_0 c_1 / c_0) / 2c_0$$

$$\cdots$$

$$I(\ell) = [x_\ell - x_0 c_\ell / c_0 - (x_1 - x_0 c_1 / c_0) c_{\ell-1} / c_0 - \cdots] / 2c_0$$

$$\cdots$$

or

$$X = C_1\, I \qquad (2.8)$$

where

$$C_1 = \begin{bmatrix} c_0 & & [] \\ c_1 & 2c_0 & \\ & & \ddots \\ c_k & 2c_{k-1} & \cdots \; 2c_0 \end{bmatrix}$$

EXAMPLE 1.

Given the polynomials

$$P(z) = p_0 z + p_1 \quad \text{and}$$

$$C(z) = c_0 z + c_1$$

Solve

$$\begin{bmatrix} 2c_0 & 2c_1 \\ c_1 & c_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 2(p_0^2 + p_1^2) \\ 2\, p_0 p_1 \end{bmatrix}$$

We find

$$x_0 = [(p_0^2 + p_1^2)c_0 - 2p_0 p_1 c_1] \; / \; (c_0^2 - c_1^2)$$

$$x_1 = [2p_0 p_1 c_0 - (p_0^2 + p_1^2)c_1] \; / \; (c_0^2 - c_1^2)$$

Substituting x_0 and x_1 into (2.7), we obtain

$$2I(1) = (c_0 x_1 - c_1 x_0) \; / \; c_0^2$$

$$= [2p_0 p_1 (c_0^2 + c_1^2) - 2(p_0^2 + p_1^2) c_0 c_1] \; / \; [c_0^2 (c_0^2 + c_1^2)]$$

Integral I(1) can be also obtained by use of residue theorem

Let

$$\Phi(z) = \frac{P(z)P(z^{-1})}{C(z)C(z^{-1})} \, z^{-2}$$

Then

$$I(1) = \sum_{|z_i|<1} \operatorname{Res}_{z\to z_i} \{ \Phi(z) \}$$

$$= \lim_{z\to 0} [z^2 \Phi(z)]' + \lim_{z\to -c_1/c_0} [(z - c_1/c_0)\Phi(z)]$$

$$= [p_0 p_1 (c_0^2 + c_1^2) - (p_0^2 + p_1^2) c_0 c_1] \; / \; [c_0^2 (c_0^2 - c_1^2)]$$

□

Substituting (2.8) into (2.5) gives a set of linear equations for $I(\ell)$

$$C\, C_1\, I = \varrho\, a + b \qquad (2.9)$$

Notice that the matrix $C\, C_1$ will be symmetric if its first row is divided by 2.

EXAMPLE 2.
Consider a second order system in the form of (2.9)

$$\begin{bmatrix} (c_0^2+c_1^2+c_2^2)/2 & c_0 c_1 + c_1 c_2 & c_0 c_2 \\ c_0 c_1 + c_1 c_2 & c_0^2+c_1^2+c_0 c_2 & c_0 c_1 \\ c_0 c_2 & c_0 c_1 & c_0^2 \end{bmatrix} \begin{bmatrix} I(0) \\ I(1) \\ I(2) \end{bmatrix} = \begin{bmatrix} (p_0^2+p_1^2+p_2^2)/2 \\ p_0 p_1 + p_1 p_2 \\ p_0 p_2 \end{bmatrix}$$

A simple nonsingular transformation $T_1\, T_2$ brings $C\, C_1$ to an upper triangular matrix

$$C\, C_1\, T_1\, T_2 = \begin{bmatrix} (c_0^2+c_1^2-c_2^2)/2 - c_0 c_1^2/(c_0+c_2) & c_0 c_1 & c_0 c_2 \\ 0 & c_0^2 + c_0 c_2 & c_0 c_1 \\ 0 & 0 & c_0^2 \end{bmatrix}$$

where

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -c_2/c_0 & -c_1/c_0 & 1 \end{bmatrix}$$

and

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ -c_1/(c_0+c_2) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then I(0), I(1) and I(2) can be obtained by back substitution.

□

An alternative approach is to reduce the calculation of I(ℓ) to a problem which involves calculation of square loss function. Recursive equations are given in the next section.

3. RECURSIVE FORMULAS FOR I(ℓ)

An efficient tool developed by Astrom (1970) for evaluating loss function can be exploited to solve (1.7). Introduce

$$C_k^*(z) = z^k C_k(z^{-1}) = c_0^k + c_1^k z + \ldots + c_k^k z^k$$

$$B_k^*(z) = z^k B_k(z^{-1}) = b_0^k + b_1^k z + \ldots + b_k^k z^k$$

and

$$Q_k(\ell,z)\, C_k(z) + R_k(\ell,z) = z^{\ell} B_k(z) \qquad (3.1)$$

where $Q_k(\ell,z)$ and $R_k(\ell,z)$ are the quotient and the remainder of the polynomial division (3.1). In this section we only give the formulas for the second term of the right hand side of (2.3) because fomulas for the first term have the same form.

Consider the complex integral

$$I_k(\ell) = \frac{1}{2\pi i} \oint \frac{z^{\ell} B_k(z) B_k(z^{-1})}{C_k(z) C_k(z^{-1})} \frac{dz}{z}$$

$$= \frac{1}{2\pi i} \oint \left[\frac{Q_k(\ell,z) B_k^*(z)}{C_k^*(z)} + \frac{R_k(\ell,z) B_k(z^{-1})}{C_k(z) C_k(z^{-1})} \right] \frac{dz}{z}$$

$$= Q_k(\ell,0)\frac{b_0}{c_0} + \frac{1}{2\pi i} \oint \frac{R_k(\ell,z) B_k(z^{-1})}{C_k(z) C_k(z^{-1})} \frac{dz}{z} \qquad (3.2)$$

Equation (3.1) is then reduced to a simple complex integral. In similar way define

$$C_{k-1}(z) = z^{-1}[C_k(z) - \alpha_k C_k^*(z)] \qquad \alpha_k = c_k^k / c_0^k$$

$$B_{k-1}(z) = z^{-1}[B_k(z) - \beta_k C_k^*(z)] \qquad \beta_k = b_k^k / c_0^k$$

and $$R_{k-1}(\ell,z) = z^{-1}[R_k(\ell,z) - \gamma_k C_k^*(z)] \qquad \gamma_k = r_k^k(\ell) / c_0^k$$

Recursive formulas for the complex integral

$$I^{0}_{k-1}(\ell) = \frac{1}{2\pi i}\oint \frac{R_{k-1}(\ell,z)B_{k-1}(z^{-1})}{C_{k-1}(z)C_{k-1}(z^{-1})}\frac{dz}{z}$$

$$= \left[\frac{1}{2\pi i}\oint \frac{R_{k-1}(\ell,z)B_{k-1}(z^{-1})}{C_{k}(z)C_{k}(z^{-1})}\frac{dz}{z}\right]\frac{1}{1-\alpha_k^2}$$

Substituting $R_{k-1}(\ell,z)$ and $B_{k-1}(z^{-1})$ into the above equality we get

$$(1-\alpha_k)I^{0}_{k-1}(\ell)=\frac{1}{2\pi i}\oint \frac{[R_k(\ell,z)-\gamma_k C^{*}_k(z)][B_k(z^{-1})-\beta_k C^{*}_k(z^{-1})]}{C_k(z)C_k(z^{-1})}\frac{dz}{z}$$

$$= I^{0}_{k}(\ell) - \frac{\beta_k}{2\pi i}\oint \frac{R_k(\ell,z)}{C^{*}_k(z)}\frac{dz}{z} - \frac{\gamma_k}{2\pi i}\oint \frac{B_k(z^{-1})}{C^{*}_k(z^{-1})}\frac{dz}{z} + \beta_k\gamma_k$$

$$= I^{0}_{k}(\ell) - \beta_k\gamma_k$$

Hence

$$I^{0}_{k}(\ell) = (1-\alpha_k^2)\, I^{0}_{k-1}(\ell) + \beta_k\gamma_k$$

$$= \frac{1}{c_0^k}\sum_{i=0}^{k}\frac{b_i^i r_i^i}{c_0^i} \qquad (3.3)$$

and

$$I_k(\ell) = Q_k(\ell,0)\frac{b_0^k}{c_0^k} + \frac{1}{c_0^k}\sum_{i=0}^{k}\frac{b_i^i r_i^i}{c_0^i} \qquad (3.4)$$

Equation (3.4) is a useful recursive formula for computing $I(\ell)$. The polynomials $Q(\ell,z)$ and $R(\ell,z)$ can be obtained in a straightforward manner from (3.1)

$$\left.\begin{aligned} Q_k(\ell;z) &= q_0 z^{\ell} + q_1 z^{\ell-1} + \ldots + q_{\ell} \\ R_k(\ell;z) &= r_1 z^{k} + r_2 z^{k-1} + \ldots + r_k \end{aligned}\right\} \qquad (3.5)$$

where

$$\left.\begin{aligned} q_0 &= b_0 / c_0 \\ q_1 &= (b_1 - q_0 c_1) / c_0 \\ &\ldots \\ q_{\ell} &= (b_{\ell} - q_0 c_{\ell} - q_1 c_{\ell-1} - \ldots - q_{\ell-1} c_1) / c_0 \\ r_1 &= b_{\ell+1} - q_0 c_{\ell+1} - q_1 c_{\ell} - \ldots - q_{\ell} c_1 \\ r_2 &= b_{\ell+2} - q_0 c_{\ell+2} - q_1 c_{\ell+1} - \ldots - q_{\ell} c_2 \\ &\ldots \\ r_k &= \quad - q_0 c_{\ell+k} - q_1 c_{\ell+k-1} - \ldots - q_{\ell} c_k \end{aligned}\right\} \qquad (3.6)$$

The coefficients of Q(ℓ;z) and R(ℓ;z) can also be obtained by using the following table

	b_0	b_1	b_2	...	b_k		
$q_0 = b_0/c_0$	c_0	c_1	c_2	...	c_k		
	$r_1(0)$	$r_2(0)$	...	$r_k(0)$			
$q_1 = r_1(0)/c_0$	c_0	c_1	...	c_{k-1}	c_k		
	$r_1(1)$	...	$r_{k-1}(1)$	$r_k(1)$			
$q_2 = r_1(1)/c_0$	c_0	...	c_{k-2}	c_{k-1}	c_k		
	.						
	.						
	$r_1(k-1)$	$r_2(k-1)$	..	$r_k(k-1)$			
$q_k = r_1(k-1)/c$	c_0	c_1	...	c_{k-1}	c_k		
	$r_1(k)$	...	$r_{k-1}(k)$	$r_k(k)$			

Table1. Long division of polynomials.

EXAMPLE 3.

We will consider the same problem as in Example 1. The integral $I(\ell)$ will be determined using (3.4).

1) Use Table 1 to determine $Q(1,z)$ and $R(1,z)$

$$q_0 = p_0/c_0 \qquad \begin{matrix} p_0 & p_1 \\ c_0 & c_1 \end{matrix}$$

$$q_1 = (p_1 - p_0 c_1/c_0)/c_0 \qquad \begin{matrix} p_1 - p_0 c_1/c_0 & \\ c_0 & c_1 \end{matrix}$$

$$-(p_1 - p_0 c_1/c_0)c_1/c_0$$

Hence $Q_1(1,z) = p_0/c_0\, z + (p_1 - p_0 c_1/c_0)/c_0$

$R_1(1,z) = -(p_1 - p_0 c_1/c_0)c_1/c_0$

2) Use another table similar to Astrom(1970), called table 2 ,to determine c_i^i , r_i^i and p_i^i .

$$\begin{array}{cc|cc|cc} c_0 & c_1 & p_0 & p_1 & 0 & -(p_1 - p_0 c_1/c_0)c_1/c_0 \\ c_1 & c_0 & c_1 & c_0 & c_1 & c_0 \\ \hline c_0 - c_1 c_1/c_0 & & p_0 - p_1 c_1/c_0 & & (p_1 - p_0 c_1/c_0)c_1 c_1/c_0 c_0 & \end{array}$$

The last row gives c_0^0 , p_0^0 and r_0^0.

3) Equation (3.4) now gives

$$I_1(1) = q_1 \frac{p_0}{c_0} + \frac{1}{c_0}\left[\frac{p_1 r_1}{c_0} + \frac{p_0^0 r_0^0}{c_0^0}\right]$$

$$= [p_0 p_1 (c_0^2 + c_1^2) - (p_0^2 + p_1^2) c_0 c_1] \,/\, [c_0^2 (c_0^2 + c_1^2)]$$

□

Summary

Summing up, we get the following algorithm for solving the factorization problem (1.3).

1) Given $A(z)$, $B(z)$ and ϱ.

2) Solve the Newton iteration equation (2.1) with the initial polynomial

$$C_0(z) = 1$$

It follows from (2.5) that

$$C\,X_1 = 2\,\varrho\,a$$

$$C\,X_2 = 2\,b$$

and

$$X = X_1 + X_2$$

It follows from (2.1) that it is sufficient to find x_i for i=0 to k-2 if the order of A(z) is k. The variable x_k can be obtained from the (k+1)th row. and x_{k-1} is then obtained by substitution.

3) X is given by I(ℓ) , see (2.8) where I(1) is given by recursive formulae (3.4). It is convenient to use Table 1 and Table 2 to solve this problem.

EXAMPLE 4.

Given $\varrho = 3$ and

$$A(z) = z^3 + 3z^2 + 3z + 3$$

$$B(z) = z^3 + 3z^2 + 3z + 3$$

Factorize P(z) by means of the program called REFAC1 listed in appendix A.

step	c_0	c_1	c_2	c_3
0	1.000000	0.000000	0.000000	0.000000
1	56.500000	84.000010	48.000000	12.000000
2	29.191400	42.015720	24.027940	6.012446
3	16.140390	21.132080	12.227490	3.099148
4	10.049080	11.136740	7.002997	1.913081
5	7.291217	7.128360	5.556308	1.719164
6	6.305313	6.154831	5.727749	1.878278
7	6.041695	6.018396	5.958360	1.981686
8	6.001110	6.000490	5.998890	1.999510
9	6.000001	6.000000	6.000000	2.000000
10	6.000000	6.000000	6.000000	2.000000
11	6.000000	6.000000	6.000000	2.000000
12	6.000000	6.000000	6.000000	2.000000

The resultant polynomial P(z) is given by

$$P(z) = 6z^3 + 6z^3 + 6z + 2$$

The number of steps required for convergence depends on the initial condition. Convergence is faster if $C_0(z)$ is closer to P(z) as is seen in the following table.

step	c_0	c_1	c_2	c_3
0	10.583000	7.937253	4.535573	1.133893
1	7.416798	6.534993	4.937915	1.473130
2	6.382465	6.171319	5.626061	1.823389
3	6.063704	6.028771	5.936308	1.971217
4	6.002530	6.001125	5.997470	1.998875
5	6.000004	6.000002	5.999996	1.999998
6	6.000000	6.000000	6.000000	2.000000
7	6.000000	6.000000	6.000000	2.000000
8	6.000000	6.000000	6.000000	2.000000
9	6.000000	6.000000	6.000000	2.000000
10	6.000000	6.000000	6.000000	2.000000
11	6.000000	6.000000	6.000000	2.000000
12	6.000000	6.000000	6.000000	2.000000

This example is not a real system , but it shows that the method has a high precision. Another result obtained by a method of elimination is given for comparison.

step	c_0	c_1	c_2	c_3
0	10.583000	7.937253	4.535573	1.133893
1	7.416800	6.534993	4.937914	1.473130
2	6.382465	6.171319	5.626060	1.823389
3	6.063703	6.028772	5.936308	1.971218
4	6.002529	6.001125	5.997470	1.998876
5	6.000006	6.000002	5.999996	1.999997
6	6.000000	6.000000	6.000000	2.000000
7	6.000002	6.000000	6.000000	1.999999
8	6.000000	5.999999	6.000001	2.000000
9	6.000001	6.000001	5.999999	2.000000
10	6.000000	6.000000	6.000001	2.000000
11	6.000001	6.000000	6.000000	2.000000
12	6.000002	6.000000	5.999999	1.999999

4. REDUCED VERSION

The disadvantage of the algorithm given in section 3 is that it requires computation time more than some other methods. This problem can be solved by a modified algorithm. Consider

$$\rho\, A(z)A(z^{-1}) + B(z)B(z^{-1}) = s_0 + s_1(z + z^{-1}) + \ldots + s_k(z^k + z^{-k})$$

$$= (s_k z^{2k} + s_{k-1} z^{2k-1} + \ldots + s_0 z^k + \ldots + s_k)\, z^{-k}$$

$$= S(z)\, z^{-k} \qquad (4.1)$$

The integral (1.7) becomes

$$I(\ell) = \frac{1}{2\pi i} \oint \frac{\rho\, A(z)A(z^{-1}) + B(z)B(z^{-1})}{C(z)C(z^{-1})} \frac{dz}{z^{\ell+1}}$$

$$= \frac{1}{2\pi i} \oint \frac{S(z)z^{-k}}{C(z)C(z^{-1})} \frac{dz}{z^{\ell+1}}$$

$$= \frac{1}{2\pi i} \oint \left[\frac{S_q(z)z^{-k}}{C(z^{-1})} + \frac{S_r(z)z^{-k}}{C(z)C(z^{-1})} \right] \frac{dz}{z^{\ell+1}}$$

$$= \frac{1}{2\pi i} \oint \frac{S_r(z)z^{-k}}{C(z)C(z^{-1})} \frac{dz}{z^{\ell+1}} \qquad (4.2)$$

where

$$S_q(z)C(z) + S_r(z) = S(z) \qquad (4.3)$$

$$S_q(z) = s_{q0} z^k + s_{q1} z^{k-1} + \ldots + s_{qk-1} z$$

$$S_r(z) = s_{r0} z^k + s_{r1} z^{k-1} + \ldots + s_{rk-1} z + s_{rk}$$

or in matrix form

$$\begin{bmatrix} c_0 & & & & & & \\ c_1 & c_0 & & & & & \\ & \ddots & \ddots & & & & \\ c_k & c_{k-1} & \cdots & c_1 & 1 & & \\ 0 & c_k & \cdots & c_2 & 0 & 1 & \\ & \ddots & \ddots & & & & \\ & & & 0 & 0 & 0 \ \cdots \ 1 & \end{bmatrix} \begin{bmatrix} s_{q0} \\ s_{q1} \\ \vdots \\ s_{r0} \\ s_{r1} \\ \vdots \\ s_{rk} \end{bmatrix} = \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_k \\ s_{k-1} \\ \vdots \\ s_0 \end{bmatrix}$$

The procedure now follows the previous one. Define

$$Q_s(\ell,z)C(z) + R_s(\ell,z) = z^{\ell} S_r(z) \qquad (4.4)$$

Replacing $R(\ell,z)$ and $B(z)$ by $R_s(\ell,z)$ and z^k in (3.2) gives

$$I_k(\ell) = [\, Q_s(\ell,0) + r_0^0 / c_0^0 \,] / c_0^k \qquad (4.5)$$

There is a general formula for r_0^0. Assume

$$R_k(\ell,z) = r_1 z^{k-1} + r_2 z^{k-2} + \ldots + r_k$$

It follows from the reduction rule that

$$R_{k-1}(\ell,z) = z^{-1}[\, R_k(\ell,z) - \gamma_k C_k^*(z) \,] \qquad \gamma_k = r_k^k(\ell) / c_0^K$$

Furthermore

$$r_0^0(\ell) = k_1 r_1(\ell) + k_2 r_2(\ell) + \ldots + k_k r_k(\ell)$$

where

$$k_1 = - c_1^1 / c_0^1$$

$$k_2 = - [\, c_2^2 + c_1^2 k \,] / c_0^2$$

$$\ldots$$

$$k_k = - [\, c_k^k + c_{k-1}^k k_1 + c_{k-2}^k k_2 + \ldots + c_1^k k_{k-1} \,] / c_0^k$$

These equations can be written as

$$r_0^0(\ell) = \sum_{i=1}^{k} k_i r_i(\ell) \qquad (4.6)$$

$$k_i = -\left(\sum_{J=1}^{i} c_J^i k_{i-J}\right) / c_0^i \qquad \text{for } i=1 \text{ to } k \qquad (4.7)$$

with $k_0 = 1$.

EXAMPLE 5.

Consider the same problem as in Example 1. We have

$$S(z) = p_0 p_1 z^2 + (p_0^2+p_1^2)z + p_0 p_1 = s_0 z^2 + s_1 z + s_0$$

Determine I(o) and I(1) by use of the recursive formulas (4.5)

1) Use a table to determine $Q_s(0,z)$, $Q_s(1,z)$, $R_s(0,z)$ and $R_s(1,z)$.

	s_0	s_1	s_0
$q_0 = s_0/c_0$	c_0	c_1	
		$s_1 - q_0 c_1$	
$q_1 = [s_1 - s_0 c_1/c_0]/c_0$		c_0	c_1
			$s_0 - q_1 c_1$
$q_2 = \{s_0 - [s_1 - s_0 c_1/c_0]c_1/c_0\}/c_0$			c_0

c_1

$-q_2 c_1$

$$Q_s(0,z) = q_0 z + q_1$$

$$Q_s(1,z) = q_0 z^2 + q_1 z + q_2$$

$$R_s(0,z) = s_0 - q_1 c_1 = r_1(0)$$

$$R_s(1,z) = -q_2 c_1 = r_1(1)$$

2) Use table 2 to determine c_i^i

c_0	c_1
c_1	c_0
$c_0 - c_1 c_1/c_0$	

3) From (4.5), we obtain

$$I(0) = \{ q_1 + [r_1(0)c_1/c_0] / [c_0 - c_1 c_1/c_0] \} / c_0$$

$$= [(p_0^2+p_1^2)c_0 - 2p_0 p_1 c_1] / [c_0(c_0^2-c_1^2)]$$

$$I(1) = \{ q_2 + [r_1(1)c_1/c_0] / [c_0 - c_1 c_1/c_0] \} / c_0$$

$$= [p_0 p_1 (c_0^2+c_1^2) - (p_0^2+p_1^2)c_0 c_1] / [c_0^2(c_0^2 - c_1^2)]$$

□

The Simplified Algorithm

A reduced algorithm is now obtained as follows

1) Multiply $S(z) = z^k [\varrho A(z)A(z^{-1}) + B(z)B(z^{-1})]$

2) Obtain $S_r(z)$ (4.3) ,$Q_s(\ell,z)$ and $R_s(\ell,z)$ (4.4).

3) Calculate $r_0^0(\ell)$ and $I(\ell)$, (4.6) and (4.5).

4) Obtain $X_J(z)$ and $C_{J+1}(z)$, (2.8).

This algorithm takes $3.5n^2 + 1.5n - 3$ multiplication and divisions if x_{n-1} and x_n are determined by back substitution, where n is the order of the system.

EXAMPLE 6.

Given $\varrho=1$ and

$$A(z) = z^3 - 1.60\, z^2 + 1.61\, z - 0.776$$

$$B(z) = z^2 - 0.95\, z + 0.2$$

Factorize P(z) by means of the program called REFAC2 listed in appendix B.

Step	c_0	c_1	c_2	c_3
0	1.000	0.000	0.000	0.000
1	5.052	-6.732	3.082	-0.776
2	3.089	-3.860	1.736	-0.455
3	2.283	-2.633	1.234	-0.370
4	1.946	-2.187	1.158	-0.395
5	1.818	-2.073	1.194	-0.425
6	1.782	-2.058	1.219	-0.435
7	1.775	-2.061	1.226	-0.437
8	1.775	-2.062	1.228	-0.437
9	1.775	-2.063	1.228	-0.437
10	1.775	-2.063	1.228	-0.437
11	1.775	-2.063	1.228	-0.437
12	1.775	-2.063	1.228	-0.437

The resultant polynomial is given by

$$P(z) = 1.775z^3 - 2.063z^2 + 1.228z - 0.437$$

The accuracy of this algorithm is not as good as with the previous algorithm but it is acceptable in practice. The result obtained by the reduced version to the same numerical example in section 3 is listed as follows.

step	c_0	c_1	c_2	c_3
0	10.583000	7.937253	4.535573	1.133893
1	7.424470	6.536308	4.936104	1.472308
2	6.351079	6.175473	5.642726	1.829136
3	6.019774	6.039279	5.977029	1.984860
4	5.994448	5.998134	6.004982	2.001781
5	6.000621	5.999140	5.998795	1.999791
6	6.000024	6.000298	6.000184	1.999992
7	5.999967	5.999946	5.999983	2.000011
8	6.000010	6.000007	6.000001	1.999997
9	5.999997	6.000000	6.000001	2.000001
10	6.000000	5.999999	6.000001	2.000000
11	6.000000	6.000000	6.000000	2.000000
12	6.000000	5.999997	6.000000	2.000000

5. REFERENCES

1.Astrom K.J. (1970): Introduction to Stochastic Control Theory. Academic Press, New York.

2.Astrom K.J. (1979): Algebraic System Theory as a Tool for Regulator Design. Report CODEN:LUTFD2/(CTFRT-7164)/1-023/ (1979)

3.Astrom K.J., Jory E.I., and Agniel R.G. (1970): A Numerical Method for the Evaluation of Complex Integrals. IEEE Trans. AC-70, 468-471 Aug. 1970.

4.Kucera V. and Vostry Z. (1976): Expanding Spectral Density into Correlation Sequence. IEEE Trans. AC-76, 592-593 Aug. 1976.

5.Kucera V. (1979): Discrete Linear Control. The Polynomial Equation Approach. Academia Prague.

6.Murthy D.N.P. (1973): Factorization of Discrete-process Spectral Matrices. IEEE Trans. IT-73, 693-696 Sept. 1973.

```
Program factorization;

{ File is called REFAC1.
  Author Z.Y.Zhou.
  Date   Apr.81.

  The program uses Newton's iteration and recursive fomulae
  of integral I (1) to perform the spectral factorization.
                -1                  -1              -1
     P(z)*P(z   ) = r*A(z)*A(z   ) + B(z)*B(z   ).

  The recursive relationship is given by
                 -1                 -1              -1
     C (z)*X (z   ) + X (z)*C (z   ) = P(z)*P(z   )
      i     i          i     i

     C    (z) = ( C (z) + X (z) ) / 2
      i+1          i       i
  where i indicates the i-th iteration.

  The solution of X(z) is obtained by

     x   = 2*c *I (i) + 2*c *I (i-1) + ... + c *I (0)
      i       0  k          1  k                i  k

  where  I (1) is an integral of function f (z) along the
          k                                    k
  unit circle
                          -1                  -1
     f (z) = P (z)*P (z   ) / ( C (z)*C (z   ) )
      k       k     k            k     k            }

const n=10;
      m=50;
      l=8;
      eps=0.001;
var a,b,at,bt,ct,cr,x:array[0..n] of real;
    c:array[0..m,0..n] of real;
    qa,qb,int,fra,frb:array[0..l] of real;
    ra,rb:array[0..l,0..n] of real;
    na,ni,la,i,j,s,p:integer;
    stab:array[0..m] of integer;
    fa,fb,fc,r:real;
    ch:char;
    filename:array[1..n] of char;

{-------------------------------------------------------------Input}
Procedure Input;
{ Input Na    -- degree of A(z);
        A[i] -- element of A(z);
        Ni    -- maximum munber of iteration;
        filename -- the name of a file for store results.  }
begin
  writeln('Input Ni (the number of iteration) and filename.');
  read(ni,filename);
  writeln('Input Na (the degree of A(z))');
  read(na);
  repeat
    writeln('Input A[i]');
    for i:=0 to na do read(a[i]);
    writeln('Input B[i]');
```

```
    for i:=0 to na do read(b[i]);
    writeln('Input r (the weighting coefficient)');
    read(r);
    writeln('Please check A[i],rewrite?');
    repeat read(ch) until ch<>' ';
  until ch<>'Y';
end; { of input }

{-----------------------------------------------------Initialize}
Procedure Initialize;
begin
  writeln(' Set initial C[z] ?');
  repeat read(ch) until ch<>' ';
  if ch='Y' then
     for i:=0 to na do read(c[0,i])
     else begin
       for i:=1 to na do c[0,i]:=0;
       c[0,0]:=1;
     end;
  la:=na-2;
  for i:=0 to ni do stab[i]:=0;
end; { of initialize }

{-----------------------------------------------------Polydivision}
Procedure Polydivision(k:integer);
begin
  qa[0]:=a[0]/c[k,0];
  qb[0]:=b[0]/c[k,0];
  for i:=1 to na do begin
      ra[0,i]:=a[i]-qa[0]*c[k,i];
      rb[0,i]:=b[i]-qb[0]*c[k,i];
  end;
  int[0]:=0;
  if la>0 then
     for s:=1 to la do begin
         qa[s]:=ra[s-1,1]/c[k,0];
         qb[s]:=rb[s-1,1]/c[k,0];
         ra[s-1,na+1]:=0;
         rb[s-1,na+1]:=0;
         for i:=1 to na do begin
             ra[s,i]:=ra[s-1,i+1]-qa[s]*c[k,i];
             rb[s,i]:=rb[s-1,i+1]-qb[s]*c[k,i];
         end;
         ra[s,0]:=0;
         rb[s,0]:=0;
     int[s]:=r*qa[s]*a[0]+qb[s]*b[0];
  end;
end;{ of polydivision }

{-----------------------------------------------------Iteration}
Procedure Iteration(k:integer);
var crm,int1:real;
    laa:integer;
begin
  for i:=0 to na do begin
      ct[i]:=c[k,i];
      at[i]:=a[i];
      bt[i]:=b[i];
  end;
  for j:=0 to na-1 do begin
      int1:=r*at[na-j]*at[na-j]+bt[na-j]*bt[na-j];
```

```
      int[0]:=int[0]+int1/ct[0];
      if la>0 then for s:=1 to la do begin
         int1:=r*at[na-j]*ra[s,na-j]+bt[na-j]*rb[s,na-j];
         int[s]:=int[s]+int1/ct[0];
      end;
      for i:=0 to na-j do cr[i]:=ct[na-j-i];
      if cr[na-j]<>0 then crm:=cr[na-j] else crm:=eps;
         fc:=ct[na-j]/crm;
         fa:=at[na-j]/crm;
         fb:=bt[na-j]/crm;
         if la>0 then for s:=1 to la do begin
            fra[s]:=ra[s,na-j]/crm;
            frb[s]:=rb[s,na-j]/crm;
         end;
      for i:=0 to na-j-1 do begin
          ct[i]:=ct[i]-fc*cr[i];
          at[i]:=at[i]-fa*cr[i];
          bt[i]:=bt[i]-fb*cr[i];
          if la>0 then for s:=1 to la do begin
             ra[s,i]:=ra[s,i]-fra[s]*cr[i];
             rb[s,i]:=rb[s,i]-frb[s]*cr[i];
          end;
      end;

      if ct[0]<0 then stab[k]:=stab[k]-1;
      if j=na-1 then begin
         int1:=r*at[na-j-1]*at[na-j-1]+bt[na-j-1]*bt[na-j-1];
         int[0]:=int[0]+int1/ct[0];
         if la>0 then for s:=1 to la do begin
            int1:=r*at[na-j-1]*ra[s,na-j-1]+bt[na-j-1]*rb[s,na-j-1];
            int[s]:=int[s]+int1/ct[0];
         end;
      end;
  end;

  if la<0 then laa:=0 else laa:=la;
  for s:=0 to laa do begin
      int[s]:=int[s]/c[k,0];
      x[s]:=c[k,s]*int[0];
      if (la>0) and (s>0) then
         for i:=1 to s do x[s]:=x[s]+2*c[k,s-i]*int[i];
  end;
  x[na]:=(2*(r*a[0]*a[na]+b[0]*b[na])-c[k,na]*x[0])/c[k,0];
  if na>1 then begin
     x[na-1]:=2*(r*(a[0]*a[na-1]+a[1]*a[na])+b[0]*b[na-1]+b[1]*b[na]);
     if na>2 then
     x[na-1]:=(x[na-1]-c[k,na-1]*x[0]-c[k,na]*x[1]-c[k,1]*x[na])/c[k,0]
     else x[1]:=(x[na-1]-c[k,1]*x[0]-c[k,1]*x[na])/(c[k,0]+c[k,2]);
  end;
end; { of iteration }

{-------------------------------------------------------------Store}
Procedure Store;
var len:integer;
    outfile:file of char;
begin
  len:=-1;
  rewrite(outfile,filename,'dat',len);
  if len=-1 then writeln('illfile');
  writeln(outfile,'Spectral Factorization:');
  write(outfile,'      ');
```

```
  for i:=0 to na do write(outfile,'    [',i:2,'] ');
  writeln(outfile); writeln(outfile);
  write(outfile,' A[z] ');
  for i:=0 to na do write(outfile,a[i]:8:4);
  writeln(outfile);
  write(outfile,' B[z] ');
  for i:=0 to na do write(outfile,b[i]:8:4);
  writeln(outfile); writeln(outfile);
  writeln(outfile,'   r= ',r:8:4);
  writeln(outfile);
  for j:=0 to ni do begin
      write(outfile,j:6);
      for i:=0 to na do write(outfile,c[j,i]:8:4);
      if stab[j]<0 then write(' unstable ');
      writeln(outfile);
  end;
  close(outfile);
end; { of store }

{=============================================Code of main program}
begin
  input;
  initialize;
  for p:=0 to ni do begin
      polydivision(p);
      iteration(p);
      for i:=0 to na do c[p+1,i]:=(c[p,i]+x[i])/2;
  end;
  store;
end. { of main program }
```

```
Program factorization;

{ File is called REFAC2.
  Author Z.Y.Zhou.
  Date   Apr.81.

  The program uses Newton's iteration and recursive fomulae
  of integral I (1) to perform the spectral factorization.
             -1                -1              -1
     P(z)*P(z  ) = r*A(z)*A(z  ) + B(z)*B(z  ).
                          -n
                 = S(z)*z

  The recursive relationship is given by
               -1                -1              -1
     C (z)*X (z  ) + X (z)*C (z  ) = P(z)*P(z  )
      i     i           i     i

     C    (z) = ( C (z) + X (z) ) / 2
      i+1          i       i
  where i indicates the i-th iteration.

     P (z) = lim C (z)
                 i
  The solution of X(z) is obtained by

     x  = 2*c *I (i) + 2*c *I (i-1) + ... + c *I (0)
      i       0  k          1  k                i  k

  where  I (1) is an integral of function f (z) along the
          k                                  k
  unit circle
                    -k                  -1
     f (z) = S (z)*z  / ( C (z)*C (z  ) )
      k       k             k     k            }

const n=10;
      m=50;
      eps=0.001;
var a,b,s,sr,x,q,int:array[0..n] of real;
    c:array[0..m,0..n] of real;
    rrs:array[0..n,0..n] of real;
    na,ni,i,j,l,p:integer;
    stab:array[0..m] of integer;
    r:real;
    ch:char;
    filename:array[1..n] of char;

{-------------------------------------------------------------Input}
Procedure Input;
{ Input Na   -- degree of A(z);
        A[i] -- element of A(z);
        Ni   -- maximum munber of iteration;
        filename -- the name of a file for store results.  }
begin
  writeln('Input Ni (the number of iteration) and filename.');
  read(ni,filename);
  writeln('Input Na (the degree of A(z))');
  read(na);
  repeat
    writeln('Input A[i]');
```

```
    for i:=0 to na do read(a[i]);
    writeln('Input B[i]');
    for i:=0 to na do read(b[i]);
    writeln('Input r (the weighting coefficient)');
    read(r);
    writeln('Please check A[i],rewrite?');
    repeat read(ch) until ch<>' ';
  until ch<>'Y';
end; { of input }

{-----------------------------------------------------Initialize}
Procedure Initialize;
begin
  writeln(' Set initial C[z] ?');
  repeat read(ch) until ch<>' ';
  if ch='Y' then
     for i:=0 to na do read(c[0,i])
     else begin
       for i:=1 to na do c[0,i]:=0;
       c[0,0]:=1;
     end;
  for i:=0 to ni do stab[i]:=0;
end; { of initialize }

{-----------------------------------------------polymultiply}
Procedure polymultiply;
var aa,bb:array[0..n] of real;
begin
  for i:=0 to na do begin
      aa[i]:=0;
      bb[i]:=0;
      for j:=i to na do begin
          aa[i]:=aa[i]+a[j-i]*a[j];
          bb[i]:=bb[i]+b[j-i]*b[j];
      end;
      s[na-i]:=r*aa[i]+bb[i];
  end;
end; { of polymultiply }

{-----------------------------------------------------polyfactor}
Procedure Polyfactor(k:integer);
var sq:array[0..n] of real;
begin
  sq[0]:=s[0]/c[k,0];
  for i:=1 to na-1 do begin
      sq[i]:=s[i];
      for j:=1 to i do
          sq[i]:=sq[i]-sq[i-j]*c[k,j];
      sq[i]:=sq[i]/c[k,0];
      write(sq[i]:8:4);
  end;
  for i:=0 to na-1 do begin
      sr[i]:=s[na-i];
      for j:=0 to na-i-1 do
          sr[i]:=sr[i]-sq[j+i]*c[k,na-j];
  end;
  sr[na]:=s[0];
end; { of polyfactor }

{-----------------------------------------------------Polydivision}
Procedure Polydivision(k:integer);
```

```
begin
  q[0]:=sr[0]/c[k,0];
  for i:=1 to na do
      rrs[0,i]:=sr[i]-q[0]*c[k,i];
  int[0]:=q[0];
  for l:=1 to na do begin
      q[l]:=rrs[l-1,1]/c[k,0];
      rrs[l-1,na+1]:=0;
      for i:=1 to na do
          rrs[l,i]:=rrs[l-1,i+1]-q[l]*c[k,i];
      int[l]:=q[l];
  end;
end;{ of polydivision }

{-------------------------------------------------------------Iteration}
Procedure Iteration(k:integer);
var cr,f,af:array[0..n] of real;
    ct:array[0..n,0..n] of real;
    int1:real;
begin
  for i:=0 to na do
      ct[0,i]:=c[k,i];
  for j:=0 to na-1 do begin
      for i:=0 to na-j do cr[i]:=ct[j,na-j-i];
      f[j]:=ct[j,na-j]/cr[na-j];
      for i:=0 to na-j-1 do
          ct[j+1,i]:=ct[j,i]-f[j]*cr[i];
      if ct[j,0]<0 then stab[k]:=stab[k]-1;
  end;

  af[1]:=-f[na-1];
  for i:=2 to na do
      for j:=1 to i-1 do
          af[i]:=-f[na-i]-af[i-j]*ct[na-i,j]/ct[na-i,0];
  for l:=0 to na do begin
      int1:=0;
      for j:=1 to na do
          int1:=int1+af[j]*rrs[l,j];
      int1:=int1/ct[na,0];
      int[l]:=int[l]+int1;
  end;
  for l:=0 to na do begin
      int[l]:=int[l]/c[k,0];
      x[l]:=c[k,l]*int[0];
      for i:=1 to l do
          x[l]:=x[l]+2*c[k,l-i]*int[i];
  end;
end; { of iteration }

{-----------------------------------------------------------------Store}
Procedure Store;
var len:integer;
    outfile:file of char;
begin
  len:=-1;
  rewrite(outfile,filename,'dat',len);
  if len=-1 then writeln('illfile');
  writeln(outfile,'Spectral Factorization:');
  write(outfile,'      ');
  for i:=0 to na do write(outfile,'   [',i:2,'] ');
  writeln(outfile); writeln(outfile);
```

```
  write(outfile,' A[z] ');
  for i:=0 to na do write(outfile,a[i]:8:4);
  writeln(outfile);
  write(outfile,' B[z] ');
  for i:=0 to na do write(outfile,b[i]:8:4);
  writeln(outfile); writeln(outfile);
  writeln(outfile,'  r= ',r:8:4);
  writeln(outfile);
  for j:=0 to ni do begin
      write(outfile,j:6);
      for i:=0 to na do write(outfile,c[j,i]:8:4);
      if stab[j]<0 then write(' unstable ');
      writeln(outfile);
  end;
  close(outfile);
end; { of store }

{=============================================Code of main program}
begin
  input;
  polymultiply;
  initialize;
  for p:=0 to ni do begin
      polyfactor(p);
      polydivision(p);
      iteration(p);
      for i:=0 to na do c[p+1,i]:=(c[p,i]+x[i])/2;
  end;
  store;
end. { of main program }
```