



Letter to the editor

A new algorithm for linear systems of the Pascal type[☆]

Xiao-Guang Lv, Ting-Zhu Huang^{*}, Zhi-Gang Ren

School of Applied Mathematics, University of Electronic Science and Technology of China, Chengdu, Sichuan, 610054, China

ARTICLE INFO

Article history:

Received 23 December 2006

Received in revised form 15 December 2007

MSC:

65F10

Keywords:

Pascal matrix

Generalized Pascal matrix

Factorization

Algorithm

Toeplitz matrix

ABSTRACT

In this paper, we give an algorithm for solving linear systems of the Pascal matrices. The method is based on the explicit factorization of the Pascal matrices. The algorithm costs no multiplications and $O(n^2)$ additions. The linear systems of the generalized Pascal matrices are also considered. Some examples are given.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

The Pascal matrix arises in applications such as in binomial expansion, Filter design [1,4], probability, combinatorics, linear algebra [9], electrical engineering [13] and order statistics [14], etc. In recent years, the linear systems of equations with coefficient matrices of Pascal type have been quietly investigated in different fields of applied linear algebra, for instance, [2,5,11,15].

For example, consider a particular solution y^* of an m th order non-homogeneous differential equation with constant coefficients of the following form:

$$a_m \frac{d^m y}{dt^m} + a_{m-1} \frac{d^{m-1} y}{dt^{m-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y = (q_0 + q_1 t + \cdots + q_n t^n) e^{at}, \quad (1)$$

where $n, m \geq 0$ and a_k and q_k represent constant coefficients.

Assume a particular solution y^* of the form $y^* = (b_0 + b_1 t + \cdots + b_n t^n) e^{at}$. Substituting y^* into (1), obtain a equation and compare the coefficients of the polynomials of the left-hand side and the right-hand side of the equation. This will allow us to determine the coefficients b_i ($i = 1, \dots, n$). In fact, computing the coefficients b_i ($i = 1, \dots, n$) of the particular solution is equal to solving the following lower triangular generalized Pascal functional systems

[☆] This research was supported by 973 Programs (2008CB317110), NSFC (10771030), the Scientific and Technological Key Project of the Chinese Education Ministry (107098), the Specialized Research Fund for the Doctoral Program of Higher Education (20070614001), Sichuan Province Project for Applied Basic Research (2008JY0052) and the Project for Academic Leader and Group of UESTC.

^{*} Corresponding author. Tel.: +86 28 83201175; fax: +86 28 83200131.

E-mail addresses: xiaoguanglv@126.com (X.-G. Lv), tzhuan@uestc.edu.cn, tingzhuang@126.com (T.-Z. Huang).

$$\begin{pmatrix} p & 0 & 0 & 0 & 0 \\ p' & p & 0 & 0 & 0 \\ p'' & 2p' & p & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ p^{(n)} & C_n^1 p^{(n-1)} & C_n^2 p^{(n-2)} & \dots & p \end{pmatrix}^T \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix}, \quad (2)$$

where $p = \sum_{k=0}^m a_k a^k$ and $p^{(i)}$ denotes the i th derivative of p with respect to a .

In this paper, we consider the solution of the linear systems of the Pascal type. A straightforward application of the Gaussian elimination to the systems of the linear equations result in an algorithm of $O(n^3)$ complexity, where n is the size of the matrix. In [7], El-Mikkawy gives the inverse of the Pascal matrix in explicit form, and obtains two algorithms for solving the linear systems of equations with coefficient matrix of Pascal type of order n . The cost of the two algorithms are $O(n^3)$ operations and $O(n^2)$ operations, respectively. Wang and Lu [12] presented a fast algorithm for solving the linear systems of equations with coefficient matrix of Pascal type of order n and the complexity is only $O(n \log n)$. The algorithm is based on the matrix decomposition. The entries of the Toeplitz matrices in the decomposition have very different magnitudes of numbers, so there can exist instability problems if the decomposition is implemented naively. An explicit factorization of the Pascal matrix is given in [8].

In this paper, we present an algorithm for linear systems of equations with coefficient matrices of Pascal type with the factorization of the Pascal matrices by the Jordan matrices. In the algorithm, we must consider computing matrix-vector products for Pascal matrices. In general, the matrix-vector multiplication algorithm costs $O(n^2)$ multiplications and $O(n^2)$ additions. But, the algorithm presented in this paper uses additions instead of multiplications. This means that we obtain an algorithm which costs $O(n^2)$ additions and no multiplications for solving the linear systems of Pascal type.

Our paper is organized as follows: In Section 2, we give some properties of Pascal matrices. In Section 3, an algorithm for solving linear systems of equations with coefficient matrices of Pascal type is presented. Finally, illustrative examples are given.

To begin with, some formal definitions are given in the following:

Definition 1 ([11]). The $n \times n$ Pascal matrix P is defined as

$$P_{ij} = \binom{i+j-2}{j-1}, \quad i, j = 1, 2, \dots, n.$$

For example, the Pascal matrix of order 6 is

$$P = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 6 & 10 & 15 & 21 \\ 1 & 4 & 10 & 20 & 35 & 56 \\ 1 & 5 & 15 & 35 & 70 & 126 \\ 1 & 6 & 21 & 56 & 126 & 252 \end{pmatrix}.$$

From the definition of the Pascal matrix, we have

- (1) The matrix P is symmetric and the matrix P is positive definite;
- (2) The determinant $\det(P) = 1$;
- (3) The Cholesky's factorization $P = LL^T$ is always possible;
- (4) The eigenvalues of P must come in reciprocal λ and $1/\lambda$.

Definition 2 ([3]). The lower triangular Pascal matrix of order n is a matrix of integers defined by P_L :

$$P_L(i, j) = \binom{i-1}{j-1}, \quad i, j = 1, 2, \dots, n,$$

where $\binom{i-1}{j-1} = 0$, if $j > i$.

For example, the lower triangular Pascal matrix of order 5 is

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}.$$

Definition 3 ([3]). Let n be a natural number and k be a nonnegative integer, we define the generalized Pascal matrix Q and the lower triangular generalized Pascal matrix Q_L of order n by

$$Q_{ij} = \binom{i+j+k-1}{j+k-1}, \quad i, j = 1, 2, \dots, n,$$

$$Q_L(i, j) = \binom{i+k-1}{j+k-1}, \quad i, j = 1, 2, \dots, n,$$

where $\binom{i+k-1}{j+k-1} = 0$, if $j > i$.

Obviously, if $k = 0$, Q is the Pascal matrix and Q_L is just the lower triangular Pascal matrix.

Further we need the $n \times n$ matrices I_n , K_n and H_n by

$$I_n = \text{diag}(1, 1, \dots, 1),$$

$$K_n(i, j) = \begin{cases} 1 & j = i, j = i - 1 \\ 0 & j > i, j < i - 1 \end{cases},$$

$$H_k = \begin{pmatrix} I_{n-k-1} & 0 \\ 0 & K_{k+1} \end{pmatrix}, \quad k = 1, 2, \dots, n-2, \quad \text{and} \quad H_{n-1} = K_n.$$

For $k = 1, \dots, n-1$, H_k is an n by n matrix.

2. Properties of Pascal matrices

Theorem 1 ([6]). If P is an n by n Pascal matrix, then P has Cholesky factorization $P = P_L P_L^T$, where P_L is the lower triangular Pascal matrix of order n .

Theorem 2 ([12]). The lower triangular Pascal matrix P_L can be decomposed as follows:

$$P_L = \text{diag}(d_1) \cdot T \cdot \text{diag}(d_2),$$

the vectors d_1 and d_2 are

$$d_1 = (0!, 1!, \dots, (n-1)!)^T, \quad d_2 = \left(\frac{1}{0!}, \frac{1}{1!}, \dots, \frac{1}{(n-1)!} \right)^T$$

and T is a lower triangular Toeplitz matrix:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{1!} & 1 & 0 & 0 & 0 \\ \frac{1}{2!} & \frac{1}{1!} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ \frac{1}{(n-1)!} & \frac{1}{(n-2)!} & \frac{1}{(n-3)!} & \cdots & 1 \end{pmatrix}.$$

Proof. Notice that the (i, j) entry of the lower triangular Pascal matrix is $P_L(i, j) = \frac{(i-1)!}{(i-j)(j-1)!}$ for $i \leq j$ and $P_L(i, j) = 0$ for $i < j$. That is, every entry in i th row of P_L has a common factor $(i-1)!$, and every entry in j th column P_L has a common factor $\frac{1}{(j-1)!}$. We can take out the common factor $(i-1)!$ of the i th row and the common factor $\frac{1}{(j-1)!}$ of the j th column. $\square$

At first, we have $(\text{diag}(d_1))^{-1} = \text{diag}(d_2)$. As well known, the inverse of the lower triangular Toeplitz matrix is also a lower triangular Toeplitz matrix. By easily computing, we obtain

$$T^{-1} = \begin{pmatrix} 1 & & & & \\ -\frac{1}{1!} & 1 & & & \\ \frac{1}{2!} & -\frac{1}{1!} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ (-1)^{n+1} \frac{1}{(n-1)!} & (-1)^{n+2} \frac{1}{(n-2)!} & (-1)^{n+3} \frac{1}{(n-3)!} & \cdots & 1 \end{pmatrix},$$

that is to say, $T^{-1} = \text{diag}(-1, 1, \dots, (-1)^n) \cdot T \cdot \text{diag}(-1, 1, \dots, (-1)^n)$. Therefore,

$$\begin{aligned} P_L^{-1} &= (\text{diag}(d_2))^{-1} \cdot T^{-1} \cdot (\text{diag}(d_1))^{-1} \\ &= \text{diag}(d_1) \cdot \text{diag}(-1, 1, \dots, (-1)^n) \cdot T \cdot \text{diag}(-1, 1, \dots, (-1)^n) \cdot \text{diag}(d_2) \\ &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot \text{diag}(d_1) \cdot T \cdot \text{diag}(d_2) \cdot \text{diag}(-1, 1, \dots, (-1)^n) \\ &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot P_L \cdot \text{diag}(-1, 1, \dots, (-1)^n). \end{aligned}$$

We just proved the following theorem.

Theorem 3. If P_L is the n by n lower triangular Pascal matrix and P the n by n Pascal matrix, then

$$\begin{aligned} P_L^{-1} &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot P_L \cdot \text{diag}(-1, 1, \dots, (-1)^n), \\ P_L^{-T} &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot P_L^T \cdot \text{diag}(-1, 1, \dots, (-1)^n), \\ P^{-1} &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot P_L^T P \cdot \text{diag}(-1, 1, \dots, (-1)^n). \end{aligned}$$

Theorem 4 ([8]). If P_L is the n by n lower triangular Pascal matrix, then the matrix P_L has the following Jordan factorization:

$$P_L = H_1 H_2 \cdots H_{n-1}.$$

For example, the lower triangular Pascal matrix of order 4 has the following factorization

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 3 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

So, by Theorem 4, we have

$$P_L^{-1} = \text{diag}(-1, 1, \dots, (-1)^n) \cdot H_1 H_2 \cdots H_{n-1} \cdot \text{diag}(-1, 1, \dots, (-1)^n).$$

Theorem 5. The lower triangular generalized Pascal matrix Q_L and the generalized Pascal matrix Q have the following properties:

$$\begin{aligned} Q_L^{-1} &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot Q_L \cdot \text{diag}(-1, 1, \dots, (-1)^n), \\ Q_L &= H_1 \cdots H_{n-2} H_{n-1}^{k+1}, \\ Q^{-1} &= P_L^{-T} Q_L^{-1}, \\ Q^{-1} &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot P_L^T Q_L \cdot \text{diag}(-1, 1, \dots, (-1)^n). \end{aligned}$$

3. Main results

At first, we consider the linear systems

$$P_L x = b,$$

where the matrix P_L is the n by n lower triangular Pascal matrix and the vector $b = (b_1, b_2, \dots, b_n)$. By Theorem 1 and Theorem 3, we have

$$x = P_L^{-1} b = \text{diag}(-1, 1, \dots, (-1)^n) \cdot H_1 H_2 \cdots H_{n-1} \cdot \text{diag}(-1, 1, \dots, (-1)^n) \cdot b.$$

So, we can easily obtain the following new algorithm for the linear systems $P_L x = b$.

Algorithm 1 (Algorithm for Solving Linear Systems with Lower Pascal Matrix).

1. Compute $x = \text{diag}(-1, 1, \dots, (-1)^n) \cdot b = (-b_1, b_2, \dots, (-1)^n b_n)^T$;
2. For $i = 1, 2, \dots, n-1$, compute $x = H_{n-i} \cdot x$;
3. Compute $x = \text{diag}(-1, 1, \dots, (-1)^n) \cdot x = (-x_1, x_2, \dots, (-1)^n x_n)^T$.

How much work does this algorithm require? In step 1 and step 3 of Algorithm 1, it only change sign of x_i for $i = 1, \dots, n$. In step 2, the matrix-vector product $x = H_{n-i} \cdot x$, there is no need of multiplication. For example, when $i = 1$, then

$$x = H_{n-1} \cdot x = \begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_1 + x_2 \\ \vdots \\ x_{n-1} + x_n \end{pmatrix}.$$

Table 1
Comparison of the computing complexity

Complexity	PASIM	LSPAS	WL	Our algorithm
Additions	$\frac{1}{6}n(n-1)(n+1)$	$n^2 - n$	$n \log_2 n + 4n - 4$	$n^2 - n$
Multiplications	$\frac{1}{6}n(n-1)(n+1)$	$n^2 - n$	$\frac{1}{2}(n \log_2 n - 2) + 4n + 1$	0

From this formula, it only needs $n - 1$ additions and no multiplication. Therefore, solving linear systems with the lower Pascal matrix costs

$$\frac{1}{2}((n-1) + \dots + 2 + 1) = \frac{1}{2}(n^2 - n)$$

additions, and it needs no multiplications.

Now we consider linear systems of the Pascal type $Px = b$, where the matrix P is the n by n Pascal matrix and the vector $b = (b_1, b_2, \dots, b_n)$. The solution:

$$\begin{aligned} x &= P_L^{-T} P_L^{-1} b \\ &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot P_L^T \cdot P_L \cdot \text{diag}(-1, 1, \dots, (-1)^n) b \\ &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot H_{n-1}^T \cdots H_2^T H_1^T H_1 H_2 \cdots H_{n-1} \cdot \text{diag}(-1, 1, \dots, (-1)^n) b. \end{aligned}$$

Therefore, to solve the linear system we may proceed as follows:

Algorithm 2 (Algorithm for Solving Linear Systems with Symmetric Pascal Type).

1. Compute $x = \text{diag}(-1, 1, \dots, (-1)^n) \cdot b = (-b_1, b_2, \dots, (-1)^n b_n)^T$;
2. For $i = 1, 2, \dots, n-1$, compute $x = H_{n-i} \cdot x$;
3. For $i = 1, 2, \dots, n-1$, compute $x = H_i^T \cdot x$;
4. Compute $x = \text{diag}(-1, 1, \dots, (-1)^n) \cdot x = (-x_1, x_2, \dots, (-1)^n x_n)^T$.

The algorithm only needs $n^2 - n$ additions.

For the linear systems with the lower triangular generalized Pascal matrix $Q_L y = b$ and the linear systems with the generalized Pascal matrix $Qz = b$, we have

$$\begin{aligned} y &= Q_L^{-1} b \\ &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot Q_L^{-1} \cdot \text{diag}(-1, 1, \dots, (-1)^n) \cdot b \\ &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot H_1 H_2 \cdots H_{n-2} H_{n-1}^{k+1} \cdot \text{diag}(-1, 1, \dots, (-1)^n) b; \\ z &= Q^{-1} b \\ &= P_L^{-T} Q_L^{-1} b \\ &= \text{diag}(-1, 1, \dots, (-1)^n) \cdot H_{n-1}^T H_{n-2}^T \cdots H_1^T \cdot H_1 \cdots H_{n-2} H_{n-1}^{k+1} \cdot \text{diag}(-1, 1, \dots, (-1)^n) b. \end{aligned}$$

The algorithms for linear systems $Q_L y = b$ and $Qz = b$ are analogous to Algorithms 1 and 2. They all only cost additions.

Example 1. Consider the following linear systems of the Pascal type $Px = b$:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & C_n^1 \\ 1 & 3 & 6 & \cdots & C_{n+1}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & C_n^1 & C_{n+1}^2 & \cdots & C_{2n-2}^{n-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} C_n^1 \\ C_{n+1}^2 \\ C_{n+2}^3 \\ \vdots \\ C_{2n-1}^n \end{pmatrix}.$$

Using Algorithm 2:

Step 1. Compute

$$x = \text{diag}(-1, 1, \dots, (-1)^n) \cdot b = (-C_n^1, C_{n+1}^2, C_{n+2}^3, \dots, (-1)^n C_{2n-1}^n)^T;$$

Step 2. For $i = 1, 2, \dots, n-1$, compute $x = H_{n-i} \cdot x$, then we get

$$x = (-C_n^1, C_n^2, -C_n^3, \dots, (-1)^n C_n^n)^T;$$

Step 3. For $i = 1, 2, \dots, n-1$, compute $x = H_i^T \cdot x$, then we get

$$x = (-1, 1, -1, \dots, (-1)^n)^T;$$

Table 2

Comparison of the computing complexity

Complexity	PASIM	LSPAS	WL	Our algorithm
Additions	35	30	36	30
Multiplications	35	5	20	0

Table 3Comparison of $\|x - x^*\|_\infty$

n	Standard	FFT algorithm	Stabilized FFT algorithm	Our method
6	0	2.2204e–16	7.1054e–15	0
12	0	5.5409e–9	4.5475e–13	0
24	0	6.7560e+6	3.7253e–9	0
48	0	3.3851e+43	0.0313	0
72	2.62144e+5	7.9105e+85	7.3400e+6	0
96	1.7592e+13	2.7288e+131	3.0786e+14	0
1000	Failure	Failure	Failure	0

Step 4. Compute $x = \text{diag}(-1, 1, \dots, (-1)^n) \cdot x = (1, 1, 1, \dots, 1)^T$.

For the Pascal linear systems of order n , we have the following comparison of the computing complexity.

In Table 1, PASIM and LSPAS are algorithms denoted in [7], respectively. WL means the algorithm in [12]. Our algorithm of solving the n by n Pascal matrix only costs $n^2 - n$ additions and no multiplications. If we use the algorithm LSPAS of [7], it costs $2n^2 - 2n$ operations for multiplications and additions. The algorithm of [12] solving the n by n Pascal matrix costs $O(n \log n)$ multiplications and $O(n \log n)$ additions.

Example 2. Solve the following linear systems of the Pascal type $Px = b$:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 6 & 10 & 15 & 21 \\ 1 & 4 & 10 & 20 & 35 & 56 \\ 1 & 5 & 15 & 35 & 70 & 126 \\ 1 & 6 & 21 & 56 & 126 & 252 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} -2 \\ -6 \\ -8 \\ -4 \\ 11 \\ 43 \end{pmatrix}.$$

The solution is $x = (1, 0, -4, 0, 1, 0)^T$. In Table 2, we compare the computing complexity when different algorithms applied to the linear systems of the Pascal matrix of order 6.

Example 3. Solve the following linear systems of the generalized Pascal type $Qz = b$:

$$\begin{pmatrix} 3 & 4 & 5 & 6 & 7 \\ 6 & 10 & 15 & 21 & 28 \\ 10 & 20 & 35 & 56 & 84 \\ 15 & 35 & 70 & 126 & 210 \\ 21 & 56 & 126 & 252 & 462 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{pmatrix} = \begin{pmatrix} 0 \\ -5 \\ -12 \\ -11 \\ 21 \end{pmatrix}.$$

where $Q_{ij} = \binom{i+j+k-1}{j+k-1}$ for $i, j = 1, 2, \dots, 5$ and $k = 2$. We use the formula

$$\begin{aligned} z &= Q^{-1}b \\ &= \text{diag}(-1, 1, \dots, (-1)^5) \cdot H_4^T \cdots H_1^T \cdot H_1 \cdots H_3 H_4^{2+1} \cdot \text{diag}(-1, 1, \dots, (-1)^5)b. \end{aligned}$$

In each step, for the matrix-vector product $x = H_{n-i} \cdot x$, there is no need for multiplications. It costs only 28 additions.

Example 4. Consider the matrix-vector product $x = P_L u$ where P_L denotes a lower triangular Pascal matrix of order n and $u = (1, 1, \dots, 1)^T$ an n -dimensional vector.

Our results from comparing $\|x - x^*\|_\infty$ are shown in Table 3, where x^* is the exact value and x is the computing value. Here we denote by “standard” the standard method computing $P_L u$ with standard matrix-vector multiplication. “FFT algorithm” means the method applied the FFT algorithm to the matrix-vector product in [12], and “stabilized FFT algorithm” means the modified FFT algorithm by using a stabilized parameter $\alpha \approx \frac{n-1}{e}$ in [10]. Our method is similar to Algorithm 1 in this paper. It only costs additions for computing the matrix-vector product. We find that cuptime of our method is less than others. For example, our method for solving the linear equations of the order 1000 cost 0.031 s. We also observe that even when the order n of the Pascal matrix is large, our method is suitable and effective. We don’t show this issue here.

Acknowledgment

The authors would like to thank the anonymous referees and Editor Professor Michael Ng very much for their constructive comments for revising this paper.

References

- [1] B. Psenicka, F. Garcia-Ugalde, A. Herrera-Camacho, The bilinear Z transform by Pascal matrix and its application in the design of digital filters, *IEEE Signal Process. Lett.* 9 (11) (2002) 368–C370.
- [2] Z. Zhi-zheng, The linear algebra of the generalized Pascal matrix, *Linear Algebr. Appl.* 271 (1997) 169–177.
- [3] Z. Zhi-zheng, Wang Tian-ming, Generalized Pascal matrices and recurrence sequences, *Linear Algebr. Appl.* 283 (1998) 289–299.
- [4] Jacek Konopacki, The Frequency transformation by matrix operation and its application in IIR filters design, *IEEE Signal Process. Lett.* 12 (1) (2005) 5–8.
- [5] M. Bayat, H. Teimoori, Pascal k -eliminated functional matrix and its property, *Linear Algebr. Appl.* 308 (2000) 65–75.
- [6] J.H. Mathews, *Numerical Methods for Mathematics, Science, and Engineering*, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1992.
- [7] M.E.A. El-Mikkawy, On solving linear syetems of the Pascal type, *Appl. Math. Comput.* 136 (2003) 195–202.
- [8] S.L. Yang, Explicit factorization of pascal matrices, *J. Math. Res. Exposition* 24 (1) (2004) 73–82.
- [9] Alan Edelman, Gilbert Strang, Pascal matrices, <http://web.mit.edu/18.06/www/Essays/pascal-work.pdf>.
- [10] X. Wang, J. Zhou, A fast eigenvalue algorithm for Pascal matrices, *Appl. Math. Comput.* 183 (2) (2006) 711–716.
- [11] L. Aceto, D. Trigiante, The matrices of Pascal and other greats, *Amer. Math. Monthly* 108 (2001) 232–C245.
- [12] X. Wang, L.Z. Lu, A fast algorithm for solving linear systems of Pascal type, *Appl. Math. Comput.* 175 (2006) 441–451.
- [13] V. Biolkova, D. Bielek, Generalized pascal matrix of first-order S-Z transforms, in: *ICECS'99 Pafos, Cyprus 1999*, pp. 929–931.
- [14] Rita Aggarwala, Michael P. Lamoureux, Inverting the Pascal matrix plus one, *American Math. Monthly* 109 (2001) 371–377.
- [15] Zhi-Hui Tang, Ramani Duraiswami and Nail Gumerov, Fast algorithm to compute matrix-vextor products for Pascal matrices. Technical Reports from UMIACS, 2004.