The Electronic Journal of Mathematics and Technology, Volume 1, Issue 1, ISSN 1933-2823

Recursive Computation of Inverses of Confluent Vandermonde Matrices

Shui-Hung Hou
mahoush@polyu.edu.hk
Department of Applied Mathematics
Hong Kong Polytechnic University, Hong Kong

Edwin Hou
hou@njit.edu
Department of Electrical and Computer Engineering
New Jersey Institute of Technology, USA

Abstract

A novel and simple recursive algorithm for inverting Vandermonde matrix and its confluent type is presented. The algorithm proposed here is suitable for both hand and machine computation. Examples are included to illustrate the proposed algorithm.

1 Introduction

Vandermonde matrices appear in many applications, such as polynomial interpolation [4, 13], digital signal processing [2], and control theory [8]. An interesting review may be found in [9].

As far as the inverse of the Vandermonde matrix V and its confluent type is concerned, a number of explicit formulas and computational schemes for the entries of V^{-1} have been given in [3, 10, 11, 12, 14]. However, an explicit recursive formula for the inversion of confluent Vandermonde matrices seems unavailable in the mathematical literature, and in linear systems as well as linear algebra textbooks. Recently, a recursive algorithm for inverting Vandermonde matrix as well as its confluent form has been reported by Hou [7]. The proposed algorithm is suitable for both numerical as well as symbolic computation. The key theory of the inversion procedure is based on the Leverrier-Faddeev method (see [1, 5, 6]), and the derivation of the

algorithm is a bit intricate and lengthy. It is the purpose of this paper to present a new yet simple proof of Hou's recursive algorithm, in a way more readily accessible for use in classroom.

The confluent Vandermonde matrix V related to the pairwise distinct zeros $\lambda_1, \lambda_2, \cdots, \lambda_r$ of the polynomial

$$p(s) = (s-\lambda_1)^{n_1}(s-\lambda_2)^{n_2}\cdots(s-\lambda_r)^{n_r}$$

with $n_1 + n_2 + \cdots + n_r = n$, is known to be

$$V = [V_1 V_2 \cdots V_r], \tag{1}$$

where the block matrix $V_k = V(\lambda_k, n_k)$ is of order $n \times n_k$, having elements $V(\lambda_k, n_k)_{ij} = \binom{i-1}{j-1}\lambda_k^{i-j}$ for $i \geq j$ and zero otherwise ($k = 1, 2, \ldots, r;\ i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n_k$). It is well known that the determinant of V is given by

$$\det V = \prod_{1 \leq i < j \leq r} (\lambda_i - \lambda_j)^{n_i n_j}.$$

$\lambda_1, \lambda_2, \ldots, \lambda_r$ being pairwise distinct, it follows that V is invertible.

In the case the zeros of $p(s)$ are simple (that is, $r = n$ and $n_1 = \cdots = n_r = 1$), we have the usual Vandermonde matrix, namely,

$$V = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \cdots & \lambda_n \\ \vdots & \vdots & & \vdots \\ \lambda_1^{n-1} & \lambda_2^{n-1} & \cdots & \lambda_n^{n-1} \end{bmatrix}.$$

More examples can be found in Section 5.

In the remainder of the paper it will be shown that the inverse of the confluent Vandermonde matrix $V = [V_1(\lambda_1, n_1)V_2(\lambda_2, n_2)\cdots V_r(\lambda_r, n_r)]$ in (1) has the form

$$V^{-1} = \begin{bmatrix} W_1(\lambda_1, n_1) \\ W_2(\lambda_2, n_2) \\ \vdots \\ W_r(\lambda_r, n_r) \end{bmatrix}, \tag{2}$$

where each block matrix $W_k(\lambda_k, n_k)$ is of order $n_k \times n$, and may be computed by means of a recursive procedure.

2 Preliminaries and notations

Let n be a given positive integer. We denote the $n \times n$ identity matrx by I_n, and let $\mathbf{e}_i$ be the i-th column vector in I_n, so that $I_n = [\mathbf{e}_1, \ldots, \mathbf{e}_n]$.

Associated with the polynomial

$$\begin{aligned} p(s) &= (s-\lambda_1)^{n_1}\cdots(s-\lambda_r)^{n_r} \\ &= s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a_n \end{aligned}$$

there is the $n \times n$ companion matrix

$$C = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \ldots & -a_1 \end{bmatrix}.$$

It is clear that

$$\begin{aligned} C\mathbf{e}_1 &= -a_n\mathbf{e}_n, \\ C\mathbf{e}_2 &= -a_{n-1}\mathbf{e}_n + \mathbf{e}_1, \\ C\mathbf{e}_3 &= -a_{n-2}\mathbf{e}_n + \mathbf{e}_2, \\ &\cdots \\ C\mathbf{e}_n &= -a_1\mathbf{e}_n + \mathbf{e}_{n-1}, \end{aligned}$$

and so

$$\begin{aligned} 0 &= C\mathbf{e}_1 + a_n\mathbf{e}_n, \\ \mathbf{e}_1 &= C\mathbf{e}_2 + a_{n-1}\mathbf{e}_n, \\ \mathbf{e}_2 &= C\mathbf{e}_3 + a_{n-2}\mathbf{e}_n, \\ &\cdots \\ \mathbf{e}_{n-1} &= C\mathbf{e}_n + a_1\mathbf{e}_n. \end{aligned} \tag{3}$$

Also associated with $\lambda_1, \ldots, \lambda_r$, there is the block diagonal matrix

$$J = \mathrm{diag}(J_1, \ldots, J_r),$$

where

$$J_k = J(\lambda_k, n_k) = \begin{bmatrix} \lambda_k & 1 & 0 & \cdots & 0 \\ 0 & \lambda_k & 1 & & \vdots \\ 0 & & \ddots & \ddots & 0 \\ \vdots & & & \lambda_k & 1 \\ 0 & \cdots & 0 & 0 & \lambda_k \end{bmatrix}$$

is the $n_k \times n_k$ Jordan block with eigenvalue λ_k. It is well known that the companion matrix C and the Jordan matrix J are related through the confluent Vandermonde matrix V by $V^{-1}CV = J$. As a result, we have

$$V^{-1}C = JV^{-1}. \tag{4}$$

3 Main results

Based on relations (3) and (4) we now proceed to derive the main results of the paper.

Premultiplying both sides of each equation in (3) by V^{-1} and making use of (4) give

$$\begin{aligned} 0 &= JV^{-1}\mathbf{e}_1 + a_n V^{-1}\mathbf{e}_n, \\ V^{-1}\mathbf{e}_1 &= JV^{-1}\mathbf{e}_2 + a_{n-1}V^{-1}\mathbf{e}_n, \\ V^{-1}\mathbf{e}_2 &= JV^{-1}\mathbf{e}_3 + a_{n-2}V^{-1}\mathbf{e}_n, \\ &\cdots \\ V^{-1}\mathbf{e}_{n-1} &= JV^{-1}\mathbf{e}_n + a_1 V^{-1}\mathbf{e}_n. \end{aligned} \tag{5}$$

Letting

$$\mathbf{h}_k = V^{-1}\mathbf{e}_{n-k+1}, \quad k = 1, \ldots, n,$$

we have, of course,

$$V^{-1} = [\mathbf{h}_n, \mathbf{h}_{n-1}, \ldots, \mathbf{h}_1].$$

Also we can rewrite (5) in terms of $\mathbf{h}_1, \cdots, \mathbf{h}_n$ as

$$\begin{aligned} 0 &= J\mathbf{h}_n + a_n\mathbf{h}_1, \\ \mathbf{h}_n &= J\mathbf{h}_{n-1} + a_{n-1}\mathbf{h}_1, \\ &\cdots \\ \mathbf{h}_3 &= J\mathbf{h}_2 + a_2\mathbf{h}_1, \\ \mathbf{h}_2 &= J\mathbf{h}_1 + a_1\mathbf{h}_1. \end{aligned} \tag{6}$$

$J = \mathrm{diag}(J_1, \ldots, J_r)$ being block diagonal, we may partition the vector $\mathbf{h}_j$ accordingly in the form

$$\mathbf{h}_j = \begin{bmatrix} \mathbf{h}_j^{(1)} \\ \vdots \\ \mathbf{h}_j^{(r)} \end{bmatrix},$$

where $\mathbf{h}_j^{(k)}$ is of size $n_k \times 1$ $(k = 1, \ldots, r; j = 1, \ldots, n)$. It is easy to see that the block matrix W_k in equation (2) is given by

$$W_k = \left[\mathbf{h}_n^{(k)}, \mathbf{h}_{n-1}^{(k)}, \ldots, \mathbf{h}_1^{(k)}\right].$$

Moreover, in view of equation (6) we have

$$\begin{aligned} 0 &= J_k\mathbf{h}_n^{(k)} + a_n\mathbf{h}_1^{(k)}, \\ \mathbf{h}_n^{(k)} &= J_k\mathbf{h}_{n-1}^{(k)} + a_{n-1}\mathbf{h}_1^{(k)}, \\ &\cdots \\ \mathbf{h}_3^{(k)} &= J_k\mathbf{h}_2^{(k)} + a_2\mathbf{h}_1^{(k)}, \\ \mathbf{h}_2^{(k)} &= J_k\mathbf{h}_1^{(k)} + a_1\mathbf{h}_1^{(k)}. \end{aligned} \tag{7}$$

Thus, in order to find the inverse V^{-1}, it is necessary that the starting vector $\mathbf{h}_1^{(k)}$ be first specified, and the vectors $\mathbf{h}_2^{(k)}, \mathbf{h}_3^{(k)}, \ldots, \mathbf{h}_n^{(k)}$ are then computed in succession in accordance with the recursive relations (7). This requisite starting vector $\mathbf{h}_1^{(k)}$ is provided in the next lemma.

Lemma 1 *Let there be given the partial fraction expansion of*

$$\frac{1}{p(s)} = \sum_{k=1}^{r} \left(\frac{K_1^{(k)}}{(s-\lambda_k)} + \frac{K_2^{(k)}}{(s-\lambda_k)^2} + \cdots + \frac{K_{n_k}^{(k)}}{(s-\lambda_k)^{n_k}} \right).$$

Then for $k = 1, \ldots, r$,

$$\mathbf{h}_1^{(k)} = [K_1^{(k)}, \ldots, K_{n_k}^{(k)}]^T.$$

Proof Let $f(t) = L^{-1}[1/p(s)]$ be the inverse Laplace transform of $1/p(s)$. Then

$$\begin{aligned} f(t) &= L^{-1}\left[\sum_{k=1}^{r} \left(\frac{K_1^{(k)}}{s-\lambda_k} + \cdots + \frac{K_{n_k-1}^{(k)}}{(s-\lambda_k)^{n_k-1}} + \frac{K_{n_k}^{(k)}}{(s-\lambda_k)^{n_k}} \right) \right] \\ &= \sum_{k=1}^{r} \left(K_1^{(k)} e^{t\lambda_k} + \cdots + K_{n_k-1}^{(k)} \frac{t^{n_k-2} e^{t\lambda_k}}{(n_k-2)!} + K_{n_k}^{(k)} \frac{t^{n_k-1} e^{t\lambda_k}}{(n_k-1)!} \right). \end{aligned}$$

Succesive differentiation of $f(t)$ at $t = 0$ gives

$$\left[f(0), f'(0), \ldots, f^{(n-1)}(0)\right]^T = V \left[K_1^{(1)}, \ldots, K_{n_1}^{(1)}, \ldots, K_1^{(r)}, \ldots, K_{n_r}^{(r)}\right]^T,$$

and so

$$V^{-1} \left[f(0), f'(0), \ldots, f^{(n-1)}(0)\right]^T = \left[K_1^{(1)}, \ldots, K_{n_1}^{(1)}, \ldots, K_1^{(r)}, \ldots, K_{n_r}^{(r)}\right]^T.$$

If we can show that

$$\left[f(0), f'(0), \ldots, f^{(n-1)}(0)\right]^T = \mathbf{e}_n,$$

then

$$\begin{aligned} \mathbf{h}_1 &= V^{-1}\mathbf{e}_n \\ &= \left[K_1^{(1)}, \ldots, K_{n_1}^{(1)}, \ldots, K_1^{(r)}, \ldots, K_{n_r}^{(r)}\right]^T, \end{aligned}$$

and whence

$$\mathbf{h}_1^{(k)} = \left[K_1^{(k)}, \ldots, K_{n_k}^{(k)}\right]^T, \quad k = 1, \ldots, r,$$

as claimed.

To this end, we expand $f(t)$ at $t = 0$ in powers of t to get

$$f(t) = \sum_{k=0}^{\infty} f^{(k)}(0) \frac{t^k}{k!} = \sum_{k=0}^{\infty} f_k \frac{t^k}{k!},$$

so that

$$1 = p(s) \cdot L[f(t)] = p(s) \cdot \sum_{k=0}^{\infty} \frac{f_k}{s^{k+1}}.$$

Comparing the coefficients of $s^{n-1}, \ldots, s^0$ on both sides of the last equation gives

$$\begin{aligned} f_0 &= 0, \\ f_1 + a_1 f_0 &= 0, \\ &\vdots \\ f_{n-2} + a_1 f_{n-3} + \cdots + a_{n-2} f_0 &= 0, \\ f_{n-1} + a_1 f_{n-2} + \cdots + a_{n-1} f_0 &= 1, \end{aligned}$$

which clearly implies

$$[f_0, f_1, \ldots, f_{n-2}, f_{n-1}]^T = [0, 0, \ldots, 0, 1]^T = \mathbf{e}_n.$$

This completes the proof of the lemma.

We are ready to state our main result.

Theorem 1 *Let $\lambda_1, \lambda_2, \ldots, \lambda_r$ be pairwise distinct zeros of the polynomial*

$$\begin{aligned} p(s) &= (s - \lambda_1)^{n_1} \cdots (s - \lambda_r)^{n_r} \\ &= s^n + a_1 s^{n-1} + \cdots + a_n \end{aligned}$$

given together with the partial fraction expansion of

$$\frac{1}{p(s)} = \sum_{k=1}^{r} \left(\frac{K_{n_k}^{(k)}}{(s - \lambda_k)^{n_k}} + \frac{K_{n_k-1}^{(k)}}{(s - \lambda_k)^{n_k-1}} + \cdots + \frac{K_1^{(k)}}{s - \lambda_k} \right).$$

For each $k \in \{1, 2, \ldots, r\}$, compute recursively n_k-dimensional vectors

$$\mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)}, \ldots, \mathbf{h}_n^{(k)}$$

by means of the following scheme:

$$\begin{aligned} \mathbf{h}_1^{(k)} &= \left[K_1^{(k)}, \ldots, K_{n_k}^{(k)} \right]^T, \\ \mathbf{h}_j^{(k)} &= J_k \mathbf{h}_{j-1}^{(k)} + a_{j-1} \mathbf{h}_{j-1}^{(k)}, \quad j = 2, \ldots, n \end{aligned}$$

terminating at

$$0 = J_k \mathbf{h}_n^{(k)} + a_n \mathbf{h}_n^{(k)}.$$

Then the inverse of the confluent Vandermonde matrix

$$V = [V(\lambda_1, n_1)V(\lambda_2, n_2)\cdots V(\lambda_r, n_r)]$$

is given by

$$V^{-1} = \begin{bmatrix} W(\lambda_1, n_1) \\ W(\lambda_2, n_2) \\ \vdots \\ W(\lambda_r, n_r) \end{bmatrix},$$

where for $k = 1, \ldots, r$ *the block matrix* $W_k = W(\lambda_k, n_k)$ *is of order* $n_k \times n$ *and*

$$W_k = \left[\mathbf{h}_n^{(k)}, \mathbf{h}_{n-1}^{(k)}, \ldots, \mathbf{h}_1^{(k)}\right].$$

It is noted that in the above scheme the vectors, $\mathbf{h}_j^{(k)}$, $j = 2, \ldots, r$, are to be calculated through matrix multiplication by J_k. We now recast the matrix computation involved, in a way to make it more suitable for hand computation as well.

Let $\mathbf{e}^{(n_k)}$ be the n_k-th column vector of the $n_k \times n_k$ identity matrix I_{n_k}. We write $J_k = \lambda_k I_{n_k} + N_k$, so that $N_k = J_k - \lambda_k I_{n_k} = J(0, n_k)$ is a nilpotent matrix of order n_k, i.e. $N_k^{n_k} = 0$ but $N_k^{n_k-1} \neq 0$.

Moreover, if the polynomial $h_j^{(k)}(s)$ is denoted by

$$h_j^{(k)}(s) = \left[s^{n_k-1}, \ldots, s, 1\right] \mathbf{h}_j^{(k)},$$

then we have

$$\mathbf{h}_j^{(k)} = h_j^{(k)}(s)\mathbf{e}^{(n_k)}\Big|_{s=N_k}.$$

Based on this representation and the nilpotency of the matrix N_k, we are led to Hou's recursive algorithm (see [7]) for inverting confluent Vandermonde matrices, which is given in the next section.

4 Algorithm ICVM

Algorithm ICVM: (Inversion of Confluent Vandermonde Matrices)

Let $\lambda_1, \lambda_2, \ldots, \lambda_r$ be pairwise distinct zeros of the polynomial

$$\begin{aligned} p(s) &= (s - \lambda_1)^{n_1} \cdots (s - \lambda_r)^{n_r} \\ &= s^n + a_1 s^{n-1} + \cdots + a_n \end{aligned}$$

given together with the partial fraction expansion of

$$\frac{1}{p(s)} = \sum_{k=1}^{r} \left(\frac{K_{n_k}^{(k)}}{(s-\lambda_k)^{n_k}} + \frac{K_{n_k-1}^{(k)}}{(s-\lambda_k)^{n_k-1}} + \cdots + \frac{K_1^{(k)}}{s-\lambda_k} \right).$$

For each $k \in \{1, 2, \ldots, r\}$, compute recursively polynomials

$$h_1^{(k)}(s), h_2^{(k)}(s), \ldots, h_n^{(k)}(s)$$

all of degree at most $n_k - 1$ by means of the following scheme:

$$\begin{aligned}
h_1^{(k)}(s) &= K_{n_k}^{(k)} + sK_{n_k-1}^{(k)} + \cdots + s^{n_k-1}K_1^{(k)}, \\
h_2^{(k)}(s) &= (\lambda_k + s)h_1^{(k)}(s) + a_1 h_1^{(k)}(s) \mod s^{n_k}, \\
h_3^{(k)}(s) &= (\lambda_k + s)h_2^{(k)}(s) + a_2 h_1^{(k)}(s) \mod s^{n_k}, \\
&\vdots \\
h_n^{(k)}(s) &= (\lambda_k + s)h_{n-1}^{(k)}(s) + a_{n-1} h_1^{(k)}(s) \mod s^{n_k},
\end{aligned}$$

terminating at

$$0 = (\lambda_k + s)h_n^{(k)}(s) + a_n h_1^{(k)}(s) \mod s^{n_k}.$$

Obtain a block matrix $W_k = W(\lambda_k, n_k)$ of order $n_k \times n$ via the equality

$$\left[\begin{array}{cccc} s^{n_k-1} & s^{n_k-2} & \cdots & 1 \end{array} \right] W(\lambda_k, n_k) = \left[\begin{array}{cccc} h_n^{(k)}(s) & h_{n-1}^{(k)}(s) & \cdots & h_1^{(k)}(s) \end{array} \right].$$

The inverse of the confluent Vandermonde matrix

$$V = [V(\lambda_1, n_1)V(\lambda_2, n_2)\cdots V(\lambda_r, n_r)]$$

may then be written as

$$V^{-1} = \left[\begin{array}{c} W(\lambda_1, n_1) \\ W(\lambda_2, n_2) \\ \vdots \\ W(\lambda_r, n_r) \end{array} \right].$$

5 Illustrative example

Here, we present an example to illustrate the recursive algorithm ICVM presented above. Let the confluent Vandermonde matrix V in (1) be given by

$$V = \begin{bmatrix} 1 & 0 & 0 & 1 \\ \lambda_1 & 1 & 0 & \lambda_2 \\ \lambda_1^2 & 2\lambda_1 & 1 & \lambda_2^2 \\ \lambda_1^3 & 3\lambda_1^2 & 3\lambda_1 & \lambda_2^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ -2 & 1 & 0 & 3 \\ 4 & -4 & 1 & 9 \\ -8 & 12 & -6 & 27 \end{bmatrix},$$

for which $\lambda_1 = -2, n_1 = 3$, and $\lambda_2 = 3, n_2 = 1$. The coefficients of the polynomial $p(s) = (s+2)^3(s-3)$ are given by $a_1 = 3, a_2 = -6, a_3 = -28$, and $a_4 = -24$. It is easy to determine the partial fraction expansion of $1/p(s)$ to be

$$\frac{1}{(s+2)^3(s-3)} = \frac{\frac{-1}{5}}{(s+2)^3} + \frac{\frac{-1}{25}}{(s+2)^2} + \frac{\frac{-1}{125}}{s+2} + \frac{\frac{1}{125}}{s-3}.$$

Let us consider first the case $\lambda_1 = -2$. Clearly

$$h_1(s) = -\frac{1}{5} - \frac{s}{25} - \frac{s^2}{125}.$$

Then

$$\begin{aligned} h_2(s) &= (-2+s)h_1(s) + 3h_1(s) \mod s^3 \\ &= -\frac{1}{5} - \frac{6s}{25} - \frac{6s^2}{125}, \\ h_3(s) &= (-2+s)h_2(s) - 6h_1(s) \mod s^3 \\ &= \frac{8}{5} + \frac{13s}{25} - \frac{12s^2}{125}, \\ h_4(s) &= (-2+s)h_3(s) - 28h_1(s) \mod s^3 \\ &= \frac{12}{5} + \frac{42s}{25} + \frac{117s^2}{125}. \end{aligned}$$

As a check of computation, we verify that

$$(-2+s)h_4(s) - 24h_1(s) \mod s^3 = \frac{117s^3}{125} \mod s^3 = 0.$$

Thus it follows from $\begin{bmatrix} s^2 & s & 1 \end{bmatrix} W_1 = \begin{bmatrix} h_4(s) & h_3(s) & h_2(s) & h_1(s) \end{bmatrix}$ that

$$W_1 = \begin{bmatrix} \frac{117}{125} & \frac{-12}{125} & \frac{-6}{125} & \frac{-1}{125} \\ \frac{42}{25} & \frac{13}{25} & \frac{-6}{25} & \frac{-1}{25} \\ \frac{12}{5} & \frac{8}{5} & \frac{-1}{5} & \frac{-1}{5} \end{bmatrix}.$$

Similarly, for $\lambda_2 = 3$ we find that

$$\begin{aligned} h_1(s) &= \frac{1}{125}, \\ h_2(s) &= (3+s)h_1(s) + 3h_1(s) \bmod s = \frac{6}{125}, \\ h_3(s) &= (3+s)h_2(s) - 6h_1(s) \bmod s = \frac{12}{125}, \\ h_4(s) &= (3+s)h_3(s) - 28h_1(s) \bmod s = \frac{8}{125}. \end{aligned}$$

and

$$(3+s)h_4(s) - 24h_1(s) \bmod s = 3 \cdot \frac{8}{125} - \frac{24}{125} = 0.$$

Then $\left[\begin{array}{c} 1 \end{array}\right] W_2 = \left[\begin{array}{cccc} h_4(s) & h_3(s) & h_2(s) & h_1(s) \end{array}\right]$ gives

$$W_2 = \left[\begin{array}{cccc} \frac{8}{125} & \frac{12}{125} & \frac{6}{125} & \frac{1}{125} \end{array}\right].$$

Finally, we have

$$V^{-1} = \left[\begin{array}{c} W_1 \\ W_2 \end{array}\right] = \left[\begin{array}{cccc} \frac{117}{125} & \frac{-12}{125} & \frac{-6}{125} & \frac{-1}{125} \\ \frac{42}{25} & \frac{13}{25} & \frac{-6}{25} & \frac{-1}{25} \\ \frac{12}{5} & \frac{8}{5} & \frac{-1}{5} & \frac{-1}{5} \\ \frac{8}{125} & \frac{12}{125} & \frac{6}{125} & \frac{1}{125} \end{array}\right].$$

All numerical results given in the example above were computed using a MATLAB version of the Algorithm ICVM. The input to the subroutine should be the eigenvalues and their multiplicities. On output, the required inverse of the given confluent Vandermonde matrix will appear. A listing of the subroutines used is provided in the Appendix.

6 Conclusions

A simple and new proof of the recursive algorithm ICVM for determining the inverses of confluent Vandermonde matrices was presented. The proposed algorithm computes one block of the inverse matrix at a time. Moreover, the simple recursive structure of the algorithm makes it suitable and easy for hand and machine computation.

7 Acknowledgments

This research is supported by the Research Committee of The Hong Kong Polytechnic University.

References

[1] D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra*, Freeman, San Francisco, 1963.

[2] H. K. Garg, *Digital Signal Processing Algorithms*, CRC Press, 1998.

[3] F. A. Graybill, *Matrices with Applications to Statistics*, second ed., Wadsworth, Belmont, Calif., 1983.

[4] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[5] S. H. Hou, A Simple Proof of the Leverrier-Faddev Characteristic Polynomial Algorithm, *SIAM Review.*, 40:706-709, 1998.

[6] S. H. Hou, On Leverrier-Faddeev Algorithm, *Proceedings of the Third Asian Technology Conference in Mathematics*, Yang et al (Eds.), Springer Verlag, 399-403, 1998.

[7] S. H. Hou, Inversion of Confluent Vandermonde Matrices, *Computers and Mathematics with Application*, No. 43, (2002), 1539-1547.

[8] T. Kailath, *Linear Systems*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1980.

[9] D. Kalman, The Generalized Vandermonde Matrix, *Math. Mag.*, 57:15-21, 1984.

[10] A. Klinger, The Vandermonde Matrix, *Amer. Math. Monthly*, 74:571-574, 1967.

[11] D. E. Knuth, *The Art of Computer Programming*, second edition, Addison-Wesley, 1973.

[12] N. Macon and A. Spitzbart, Inverses of Vandermonde Matrices, *Amer. Math. Monthly*, 65:95-100, 1958.

[13] C. Pozrikidis, *Numerical Computation in Science and Engineering*, Oxford University Press, 1998.

[14] J. F. Traub, Associated polynomials and uniform methods for the solution of linear problems. *SIAM Review*, 8:277-301, 1966.

8 Appendix (Matlab program for Algorithm ICVM)

Algorithm ICVM was implemented using Matlab version 5.3. The implementation given below consists of two subroutines *cvander* and *icvm*.

- *cvander* returns confluent Vandermonde matrix.
- *icvm* returns the inverse confluent Vandermonde matrix.

Both subroutines requires two input arguments, namely *lambda* and *n*:

- *lambda* is an array of pairwise distinct eigenvalues, $[\lambda_1, ..., \lambda_r]$.
- n is the array of the corresponding multiplicities of the eigenvalues, $[n_1, ..., n_r]$.

The user should include the subroutines as individual files within a directory. A calling program that specifies the appropriate input should also be provided. The code given below are for reference and classroom use only. Note that the *residue* command is used to determine the partial fraction expansion of $1/p(s)$. The poles obtained by *residue* may not be identical to the original input eigenvalues, and this may cause the routine to fail.

```
------------------------ icvm.m ------------------------

function v = icvm(lambda, n)
% IVCM Inversion of Confluent Vandermonde Matrix
% v = ICVM(lambda, n) finds the inverse of a confluent
%                     Vandermonde matrix.
% lambda = eigenvalues (vector)
% n = multiplicity of the eigenvalues (vector)

% build vector of eigenvalues
ei = [];
for ii=1:length(n)
   ei = [ei ,lambda(ii) * ones(1, n(ii))];
end

% x = coefficients of characteristic polynomial
x = poly(ei);
a = x(2: sum(n)+1);

% k = coefficients of the partial fraction expansion
[k p]= residue([1], x);
k = k(length(k):-1:1).';
p = p(length(p):-1:1).';

% residue arranges the coefficients of the partial fraction
% expansion in order of absolute magnitude of the poles.
% They are reordered according to the original lambda.
tk = [];
for ii=1:length(n)
```

```
   for jj=1:length(p)
      % poles calculated by residue are not identical to
      % original lambda.
      if abs(lambda(ii) - p(jj)) < 10^-5
         tk = [tk , k(jj:jj+n(ii)-1)];
         break
      end
   end
end
k = tk;

% calculate inverse Vandermonde matrix
cn = cumsum(n);
v = [];
s = 1;
for ii=1:length(n)
   h = k(s:cn(ii));
   h1 = h;
   w = [];
   for jj=1:sum(n)
      w = [ h(length(h):-1:1).', w];
      h = (lambda(ii) * h) + a(jj)* h1 + [0, h(1:length(h)-1)];
   end
   s = cn(ii)+1;
   v = [v; w];
end

------------------------- cvander.m -------------------------

function v = cvander(lambda, n)
% CVANDER confluent Vandermonde matrix
% v = CVANDER(lambda, n) finds the confluent Vandermonde matrix.
% lambda = eigenvalves (vector)
% n = multiplicities of eigenvalues (vector)

v =[];
for kk=1:length(n)
   w = [];
   for ii=1:sum(n)
      for jj=1:n(kk)
         if (ii-jj < 0)
            w(ii,jj) = 0;
         else
            x = prod(1:ii-1);
            y = prod(1:jj-1);
            z = prod(1:ii-jj);
            w(ii,jj) = (x/(y*z))*lambda(kk)^(ii-jj);
         end
      end
   end
   v = [v, w];
end

------------------------------------------------------------
```

The following is the output from MATLAB displaying the confluent Vandermonde ma-

trix (v) and its inverse (vi) with the following eigenvalues: $(\lambda_1, n_1) = (-1, 3), (\lambda_2, n_2) = (-2, 2), (\lambda_3, n_3) = (-3, 1)$. The results are checked to ensure $v * vi = vi * v = I$.

```
EDU>> v = cvander([-1 -2 -3],[3 2 1])

v =

     1     0     0     1     0     1
    -1     1     0    -2     1    -3
     1    -2     1     4    -4     9
    -1     3    -3    -8    12   -27
     1    -4     6    16   -32    81
    -1     5   -10   -32    80  -243

EDU>> vi = icvm([-1 -2 -3],[3 2 1])

vi =

   16.5000   58.0000   83.1250   56.3750   17.8750    2.1250
   -9.0000  -36.0000  -52.2500  -34.7500  -10.7500   -1.2500
    6.0000   20.0000   25.5000   15.5000    4.5000    0.5000
  -15.0000  -56.0000  -80.0000  -54.0000  -17.0000   -2.0000
   -6.0000  -23.0000  -34.0000  -24.0000   -8.0000   -1.0000
   -0.5000   -2.0000   -3.1250   -2.3750   -0.8750   -0.1250


EDU>> v * vi

ans =

    1.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000
    0.0000    1.0000    0.0000    0.0000    0.0000    0.0000
   -0.0000   -0.0000    1.0000   -0.0000   -0.0000   -0.0000
    0.0000    0.0000    0.0000    1.0000    0.0000    0.0000
   -0.0000   -0.0000   -0.0000   -0.0000    1.0000   -0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    1.0000

EDU>> vi * v

ans =

    1.0000    0.0000   -0.0000   -0.0000    0.0000    0
    0.0000    1.0000    0.0000    0.0000   -0.0000    0.0000
   -0.0000    0.0000    1.0000   -0.0000    0.0000   -0.0000
   -0.0000    0.0000   -0.0000    1.0000    0.0000   -0.0000
   -0.0000    0.0000   -0.0000   -0.0000    1.0000   -0.0000
    0.0000   -0.0000    0.0000    0.0000   -0.0000    1.0000

EDU>>
```