A DETERMINANT THEOREM WITH APPLICATIONS TO PARALLEL ALGORITHMS

Don Heller

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa.

March, 1973

This work was supported in part by the National Science Foundation under Grant GJ-3211 and by the Office of Naval Research under Contract N00014-67-A-0314-0010, NR044-422.

ABSTRACT

We state and prove an expansion theorem for the determinant of any Hessenberg matrix. The expansion is expressed as a vector-matrix-vector product which can be efficiently evaluated on a parallel machine. We consider the computation of the first N terms of a sequence defined by a general linear recurrence. On a sequential machine this problem is $O(N^2)$, with N processors it is $O(N)$, and with $O(N^4)$ processors it is $O(\log^2 N)$ using our expansion. Other applications include locating roots of analytic functions and proving doubling formulas for linear recurrences with constant coefficients.

1. Introduction

In this paper we prove a general expansion formula for the determinant of a Hessenberg (almost triangular) matrix, and show how it may be used to design algorithms for parallel computers. We analyze the algorithms for two types of parallel machines: Single Instruction Stream Multiple Data Stream (SIMD) and Multiple Instruction Stream Multiple Data Stream (MIMD). The total time to execute an algorithm is estimated by counting arithmetic steps; we neglect overhead costs.

The major application of the expansion theorem is to the following problem: Compute the first N terms of a sequence given by a general linear recurrence

$$y(i) = 0,\ i < 0;\quad y(i) = \sum_{j=0}^{i-1} A(i-1,j)\ y(j) + H(i-1),\ i \geq 0.$$

On a sequential computer this problem is $O(N^2)$. We will show that on an MIMD computer with $O(N^4)$ processors this problem is $O(\log^2 N)$, and that on an SIMD machine with N processors the problem is $O(N)$. The speedup, defined as sequential time/parallel time, is then $(N/\log N)^2$ and N, respectively. Although the MIMD algorithm is faster under the assumption that we have sufficiently many processors, it is not practical for use in current or planned computer systems.

Other applications of the determinant theorem include locating roots of analytic functions and proving doubling formulas for linear recurrences with constant coefficients.

2. Hessenberg Determinants

In this section we prove the main theorem of this paper, a recursive expansion formula for the determinant of a lower Hessenberg matrix ($M(i,j) = 0$ if $j > i+1$).

We first introduce the necessary notation. For any $n \times n$ matrix M, let $M[r:s;\ t:u]$, $1 \le r \le s \le n$, $1 \le t \le u \le n$, be the submatrix of M consisting of rows r through s and columns t through u. Define the $n \times n$ matrix K_M by

$$(2.1)\quad K_M(r,s) = \begin{cases} \det(M[r:s;r:s]) & \text{if } 1 \le r \le s \le n \\ 1 \text{ if } 2 \le r \le n, \quad s = r-1 \\ 0 \text{ otherwise.} \end{cases}$$

As a notational convenience, we define $K_M(1,0) = K_M(n+1,n) = 1$, and $M[r:s;\ t:u] = 0$ if $r > s$ or $t > u$.

The subscript M will be omitted when no ambiguity will result.

It should be observed that K_M is an upper Hessenberg matrix whose main diagonal agrees with that of M.

Theorem 2.1. If M is an $n \times n$ lower Hessenberg matrix, and if $1 \le r \le s \le t \le n$, then

$$(2.2)\quad K_M(r,t) = K_M(r,s)K_M(s+1,t) + \sum_{i=s+1}^{t} \sum_{j=r}^{s} M(i,j)\, K_M(r,j-1)\, K_M(i+1,t) \prod_{k=j}^{i-1} (-M(k,k+1)).$$

Remarks: In applications we will be primarily interested in $K_M(1,n) = \det(M)$. The above expansion is a doubling formula, as may be seen by considering $s = r+i-1$, $t = r+2i-1$. To find $K(r,r+2i-1)$, a $2i \times 2i$ determinant, requires

the determinants of lower Hessenberg matrices no larger than $i \times i$.

The theorem could be proven by applying the Laplace expansion to the first s-r+1 rows of $K_M(r,t)$, but the resulting summation is not easily seen to be (2.2). We hope to make this fact transparent with the following proof, prefixed by two lemmas.

Lemma 2.1. If M is an $n \times n$ lower Hessenberg matrix and if $1 \le r \le s \le n$, then

$$(2.3)\quad K(r,s) = \sum_{i=r}^{s} M(i,r)\, K(i+1,s) \prod_{k=r}^{i-1} (-M(k,k+1))$$

and

$$(2.4)\quad K(r,s) = \sum_{i=r}^{s} M(s,i)\, K(r,i-1) \prod_{k=i}^{s-1} (-M(k,k+1)).$$

Proof. To show (2.3), expand $K(r,s) = \det(M[r:s;r:s])$ along its first column. To show (2.4), expand along the last row.

Lemma 2.2. If $1 \le r \le s < s+1 \le t \le n$, then

$$\begin{aligned}(2.5)\quad & K(r,s)\, K(s+1,t) \\ &\quad + \sum_{i=s+1}^{t} \sum_{j=r}^{s} M(i,j)\, K(r,j-1)\, K(i+1,t) \prod_{k=j}^{i-1} (-M(k,k+1)) \\ &= K(r,s+1)\, K(s+2,t) \\ &\quad + \sum_{i=s+2}^{t} \sum_{j=r}^{s+1} M(i,j)\, K(r,j-1)\, K(i+1,t) \prod_{k=j}^{i-1} (-M(k,k+1)).\end{aligned}$$

Proof. In the lefthand side of (2.5), replace K(s+1,t) with its expansion from (2.3), and combine the summations to obtain

$$\begin{aligned}\text{LHS} &= K(r,s)\Big[\sum_{i=s+1}^{t} M(i,s+1)\, K(i+1,t) \prod_{k=s+1}^{i-1} (-M(k,k+1))\Big] \\ &\quad + \sum_{i=s+1}^{t} \sum_{j=r}^{s} M(i,j)\, K(r,j-1)\, K(i+1,t) \prod_{k=j}^{i-1} (-M(k,k+1)) \\ &= \sum_{i=s+1}^{t} \sum_{j=r}^{s+1} M(i,j)\, K(r,j-1)\, K(i+1,t) \prod_{k=j}^{i-1} (-M(k,k+1)).\end{aligned}$$

This double summation may be rewritten as

$$\sum_{j=r}^{s+1} M(s+1,j)\, K(r,j-1)\, K(s+2,t) \prod_{k=j}^{s} (-M(k,k+1)) + \sum_{i=s+2}^{t} \sum_{j=r}^{s+1} M(i,j)\, K(r,j-1)\, K(i+1,t) \prod_{k=j}^{i-1} (-M(k,k+1)).$$

Now use (2.3) to reduce the first summation to $K(r,s+1)\, K(s+2,t)$, yielding the result.

Proof of Theorem 2.1. If $s=t$, the result is trivial, so assume that $t \geq s+1$. Now apply (2.5) as many times as possible to the righthand side of (2.2), reducing it to

$$K(r,t-1)\, K(t,t) + \sum_{i=t}^{t} \sum_{j=r}^{t-1} M(i,j)\, K(r,j-1)\, K(i+1,t) \prod_{k=j}^{i-1} (-M(k,k+1)).$$

Since $K(t,t) = M(t,t)$ and $K(t+1,t) = 1$, this becomes

$$\sum_{j=r}^{t} M(t,j)\, K(r,j-1) \prod_{k=j}^{t-1} (-M(k,k+1))$$

which is $K(r,t)$ by (2.4). QED

Corollary 2.1. If $1 \leq r \leq s \leq t \leq n$ and M is tridiagonal, then

$$(2.6)\quad K(r,t) = K(r,s)\, K(s+1,t) - M(s+1,s)\, M(s,s+1)\, K(r,s-1)\, K(s+2,t).$$

This corollary was given by Sylvester [Sy 1853a], [Sy 1853b], and by Euler [Eu 1764] for a special case.

For a special class of lower Hessenberg matrices, Theorem 2.1 takes a particularly elegant form.

Definition. The lower Hessenberg matrix M is normalized if $M(j,j+1) = -1$ for $1 \le j \le n-1$.

Theorem 2.2. If M is normalized, and if $1 \le r \le s \le t \le n$, then

$$(2.7)\quad K(r,t) = K(r,s)\,K(s+1,t) + K[r:r;r-1:s-1](M[s+1:t;r:s])^{T}K[s+2:t+1;t:t]$$

and

$$(2.8)\quad K(r,t) = K[r:r;r-1:s](M[s+1:t;r:s+1])^{T}\,K[s+2:t+1;t:t]$$

Proof. Expansion (2.7) is a restatement of (2.2). Expansion (2.8) results from applying (2.3) to K(s+1,t) in (2.2). QED

Actually, it is possible to deal only with normalized matrices, as may be seen by defining the nxn matrix M' by

$$M'(i,j) = M(i,j) \prod_{k=j}^{i-1} (-M(k,k+1)),\quad 1 \le j \le i \le n,$$

$$M'(i,j) = -1,\quad 1 \le i \le n-1,\ j = i+1,$$

$$M'(i,j) = 0,\quad 1 \le i \le n-1,\ i+1 < j \le n.$$

It then follows from (2.3) and (2.4) that $K_{M'} = K_{M}$.

We now consider the case where M is a normalized band matrix.

Theorem 2.3. If M is normalized, $M(i,j) = 0$ for $i-j \ge k$, and if $1 \le r \le s \le t \le n$, then

$$(2.9)\quad K(r,t) = K(r,s)\ K(s+1,t) + \sum_{i=1}^{p} 2^{\,i-1}\ M(s+i-j,s-j)\ K(r,s-j-1)\ K(s+i-j+1,t)$$

where $p = \min(k-1, t-r)$

$q_i = \max(0,\ i-(s-r+1))$.

Proof. Note that there are only k+1 diagonals that are non-zero. The formula may then be constructed from (2.2). QED

Finally, we note that the definitions and theorems of this section may be altered slightly to provide a similar treatment for the permanent of a Hessenberg matrix. In (2.1), change "det" to "per" and in all other places change "-M(k,k+1)" to "M(k,k+1)".

3. Application to Calculating Linear Recurrences on Parallel Computers

We now show how the results of Section 2 may be used to solve a general initial value linear recurrence problem. We also indicate methods of solving boundary value problems and k^{th} order linear recurrences.

Algorithms for the three basic types of machines, sequential, SIMD and MIMD, are given and analyzed. In the general recurrence problem a speedup of N/2 is possible for SIMD machines, and $(N/\log N)^2$ for MIMD machines. For k^{th} order recurrences the speedup is N/logN for each type.

Suppose we are given a function H defined on Z, the set of integers, and a function A defined on Z × Z, with $A(i,i+1) \neq 0$ for $-1 \leq i$. Then there is a unique sequence y defined on Z such that

$$
\begin{aligned}
& y(i) = 0 \quad \text{if } i < 0, \\
(3.1)\quad & A(-1,0)\, y(0) = H(-1), \\
& A(i-1,i)\, y(i) = \sum_{j=0}^{i-1} A(i-1,j)\, y(j) + H(i-1) \quad \text{if } i > 0.
\end{aligned}
$$

As may be seen, there is no loss of generality in assuming that $A(i,i+1) = 1$ for $-1 \leq i$, and we will do so.

Actually, (3.1) defines a sequential algorithm to calculate y(0),...,y(N).

```
comment Sequential algorithm;
   y(0) := H(-1);
   for i := 1 step 1 until N do
      begin Sum := H(i-1);
         for j := 0 step 1 until i-1 do
           Sum := Sum + A(i-1,j)*y(j);
         y(i) := Sum
      end;
```

This requires $N(N+1)/2$ multiplications and an equal number of additions, so our problem is $O(N^2)$ for a one processor machine.

For an SIMD machine with N processors we can easily define an algorithm which requires 2N arithmetic steps. This is a column-wise algorithm; the previous one is row-wise.

The notation used here is a pseudo-Algol language due to Stone [St73] in which the notation $(r \le j \le s)$ after a statement means that the statement should be executed in parallel for all values of j in the interval.

```
comment parallel algorithm 1 (SIMD);
   y(i) := H(i-1), (0 ≤ i ≤ N);
   for j := 0 step 1 until N-1 do
      y(i) := y(i) + A(i-1,j) * y(j),
                 (j+1 ≤ i ≤ N);
```

This requires N multiplications and an equal number of additions, so the speedup is $N^2/2N = N/2$, using this simple technique.

We now show how to achieve a speedup of $(N/\log N)^2$ for MIMD machines with sufficiently many processors.

Define the $(N+1) \times (N+1)$ matrix B by

$$
\begin{aligned}
&B(i,1) = H(i-2), \quad 1 \le i \le N+1,\\
&B(i,j) = A(i-2, j-2), \quad 2 \le j \le i \le N+1,\\
&B(i,i+1) = -1, \quad 1 \le i \le N,\\
&B(i,j) = 0 \text{ otherwise.}
\end{aligned}
\tag{3.2}
$$

B is a normalized lower Hessenberg matrix, so we may apply the results of Section 2. In particular,

Theorem 3.1. For $0 \le i \le N$, $y(i) = K_B(1,i+1)$.

Proof. Using (2.4) and (3.2) we have

$$(3.3)\quad K_B(1,i+1) = \sum_{j=0}^{i-1} A(i-1,j)\, K_B(1,j+1) + H(i-1).$$

For $i = 0$, $K_B(1,i+1) = H(-1) = y(0)$. By the uniqueness of y and (3.3) we have the desired result.

Corollary 3.1. If $H(i) = 0$ for $0 \le i \le N-1$, then $y(i) = y(0)K_B(2,i+1)$.

Theorem 3.1 may also be proven by applying Cramer's rule to the triangular linear system

$$y(i) - \sum_{j=0}^{i-1} A(i-1,j)\, y(j) = H(i-1), \quad 0 \le i \le N,$$

which was apparently first done by Scherk in 1825 [Mu23, vol. 1]. Viewed as a problem in differences, the corollary was first stated by Sylvester [Syl853a], [Syl862] for the case $y(0) = 1$.

We can now use Theorems 2.2 and 3.1 to define a parallel algorithm to compute $y(0),\ldots,y(N)$ for $N = 2^{n+1}-1$. Since B is normalized, we have from (2.8),

$$(3.4)\quad K_B(r,t) = K_B[r:r;\ r-1:s-1](B[s+1:t;r:s+1])^T K_B[s+2:t+1;t:t]$$

Define Q(r,s,t) to be a procedure which computes $K_B(r,t)$ by (3.4).

```
comment Parallel algorithm 2 (MIMD).
   y(j) = K(1,j+1), 0 ≤ j ≤ N;
  K(r,r-1) := 1, (2 ≤ r ≤ N+1);
  K(r,r) := B(r,r), (1 ≤ r ≤ N+1);
  K(r,r+1) := B(r,r)*B(r+1,r+1) + B(r+1,r),
                (1 ≤ r ≤ N);
  for i := 1 step i+1 until N-1 do
    K(r,r+i+j) := Q(r,r+i,r+i+j), (1 ≤ j ≤ i+1),
                (1 ≤ r ≤ N+1-i-j);
```

It is easily verified that this algorithm always provides enough information to proceed to the next loop. After n loops we have computed $K(1,j+1)$ for $0 \le j \le 2^{n+1}-1 = N$. To compute $Q(r,r+i,r+i+j)$ requires two multiplications and $\lceil \log_2(i+2) \rceil + \lceil \log_2 j \rceil$ additions. This is greatest when $j = i+1$.

The total number of multiplication steps for the loop is 2n, and the total number of addition steps is

$$\sum_{k=1}^{n} \left(\lceil \log_2((2^k-1)+2) \rceil + \lceil \log_2((2^k-1)+1) \rceil \right)$$

$$= \sum_{k=1}^{n} (2k+1) = n^2+2n.$$

Thus the total number of arithmetic steps is $n^2+4n+2 = 0(\log^2 N)$, which gives a speedup of $(N/\log N)^2$ over the standard serial algorithm. This requires $O(N^4)$ processors, and so is not intended to be practical.

It should be observed that the above algorithm generates more information than is necessary. For instance, the final loop ($i = 2^n-1$) may be shortened to

```
K(1,1+i+j) := Q(1,1+i,1+i+j),  (1 ≤ j ≤ i+1);
```

This does not, however, reduce the time estimate for MIMD machines. Also note that B and K_B may be stored in the upper and lower triangular portions, respectively, of an (N+1) x (N+1) matrix.

We summarize our results as a theorem.

Theorem 3.2. The first N terms of the sequence defined by (3.1) may be computed in $O(\log^2 N)$ steps on an MIMD machine with sufficienctly many processors. This provides a speedup of $(N/\log N)^2$ over serial machines.

Parallel algorithm 2 cannot be efficiently adapted to SIMD machines. For instance, the modification

```
for i := 1 step i+1 until N-1 do
  for j := 1 step 1 until i+1 do
    K(r,r+i+j) := Q(r,r+i,r+i+j),
        (1 ≤ r ≤ N+1-i-j);
```

is O(N log N).

We now consider an important special case of (3.1), a homogeneous k^{th} order sequence, $k \geq 1$.

$$(3.5)\quad y(i) = \begin{cases} 0 \text{ if } i < 0, \\ H(i-1) \text{ if } i = 0, \\ \sum_{j=i-k}^{i-1} A(i-1,j)\, y(j) + H(i-1), \text{ if } i > 0, \\ H(i-1) = 0 \text{ if } i \geq k. \end{cases}$$

The usual formulation gives $y(i) = H(i-1)$ for $0 \leq i \leq k-1$, which is included in (3.5).

The sequential algorithm requires O(N) arithmetic operations, as do the row-wise and column-wise SIMD algorithms, so there is only a constant speedup. However, there are a variety of techniques which do obtain N/logN speedup on SIMD machines. (cf. [Ko72], [St73].)

Since B is a band matrix for this problem, we may use Theorem 2.3 to compute Q(r,r+i, r+i+j). On an MIMD machine this requires at most two

multiplications and $\lceil \log_2(1+(k-1)k/2) \rceil = a_k$ additions, for a total of O(logN) operations for parallel algorithm 2. Thus the speedup is N/logN over serial machines. Again, it is possible to purge the loop of unnecessary computations, reducing the number of processors, but not the total time.

Inhomogeneous k^{th} order sequences may also be easily treated via parallel algorithm 2, since

$$(3.6)\quad y(i) = K_B(1,i+1) = \sum_{j=1}^{i+1} B(j,1)\, K(j+1,\, i+1)$$

$$= \sum_{j=0}^{i} H(j-1)\, K_B(j+2,\, i+1).$$

B[2:N+1;2:N+1] is a band matrix, so we may find y(0),...,y(N) in 2n+1 multiplications and $a_k n + \lceil \log_2 k \rceil$ additions. This is again O(logN), a speedup of N/logN.

It is also possible to solve boundary value problems using (3.6). This involves solving an additional system of linear equations to find the appropriate starting values of y.

4. Other Applications

We first consider some classical results concerning power series and analytic functions and show how the results of Sections 2 and 3 may be applied.

Theorem 4.1. If $f(x) = \sum_{i=0}^{\infty} a_i x^i$, $a_0 \neq 1$, $g(x) = \sum_{i=0}^{\infty} b_i x^i$, $b_0 = 1$, and $h(x) = g(x)/f(x) = \sum_{i=0}^{\infty} c_i x^i$, then $c_0 \neq 1$ and for $r > 0$,

$$(4.1)\quad c_r = \begin{vmatrix} b_1 - a_1 & & & \\ b_2 - a_2 & -a_1 & & \\ \vdots & \ddots & \ddots & \\ b_r - a_r & & r-1 & -a_1 \end{vmatrix}$$

Moreover, $c_0, \ldots, c_n$ can be computed in $O(\log N)$ steps on a parallel computer with sufficiently many processors.

Proof. Since $h(x)f(x) = g(x)$, we have the well-known relation

$$a_0 c_0 = b_0, \qquad a_0 c_i + \sum_{j=1}^{i} a_j c_{i-j} = b_i, \quad i \geq 1.$$

That is, $c_0 = 1/a_0$, $c_i = \frac{1}{a_0}\left[\sum_{j=1}^{i-1} (-a_j) c_{i-j} + (b_i - a_i)\right]$, $i \geq 1$. The result follows from Theorems 3.1 and 3.2.

Konig's Theorem rHo701 state?? that a convergent sequence of approximations to the smallest real root of the analytic function- $f(x) = \sum_{i=0}^{\infty} a_i x^i$, a_0

is given by

$$r_i = c_{i-1}/c_i, \text{ where } 1/f(x) = \sum_{i=0}^{\infty} c_i x^i.$$

The sequence r_i was first given by Furstenau in 1860 [Mu23, vol. 3], and by Bernoulli in 1728 for the special case where f is a polynomial. It is seen from Theorem 4.1 that the first N Furstenau approximations may be computed in $O(\log^2 N)$ steps on a parallel machine with sufficiently many processors.

We now examine some properties of recurrences with constant coefficients, which are of interest in number theoretic applications as well as the above root finding techniques. The most familiar is the second order recurrence $F_n = F_{n-1} + F_{n-2}$, $n \geq 2$, which defines the Fibonacci sequence for $F_0 = 0$, $F_1 = 1$. Its well-known doubling formula $F_{m+n} = F_{m+1}F_n + F_m F_{n-1}$ may be generalized in the following way.

Suppose y is defined by

$$y(i) = \begin{cases} 0 & \text{if } i < 0, \\ y_i & \text{if } 0 \leq i \leq k-1, \\ \sum_{j=0}^{i-1} b_{i-1-j}\, y(j) & \text{if } i \geq k. \end{cases} \tag{4.2}$$

That is, $A(i-1,j) = b_{i-1-j}$ for $i \geq k$, $0 \leq j \leq i-1$, $A(i-1,j) = 0$ for $0 \leq i \leq k-1$, $0 \leq j \leq i-1$, $H(i-1) = y_i$ for $0 \leq i \leq k-1$, and $H(i-1) = 0$ for $k \leq i$.

Let V_i be defined by

$$(4.3)\quad V_i = \begin{cases} 0 & \text{if } i \le 0 \\ 1 & \text{if } i = 1 \\ \sum_{j=0}^{i-1} b_{i-1-j}\, V_j & \text{if } i \ge 1. \end{cases}$$

Lemma 4.1. If $i \ge 1$ then $V_i = K_B(r, r+i-2)$ for any $r \ge k+1$, where B is defined by (3.2).

The proof is immediate from (4.3).

Theorem 4.2. If $n \ge k$ then

$$(4.4)\quad y(n) = y_{k-1}\, V_{n-k+2} + \sum_{j=0}^{k-2} y_j \sum_{i=k}^{n} b_{i-1-j}\, V_{n+1-j}.$$

If $m \ge 1$ and $n \ge 1$, then

$$(4.5)\quad V_{m+n} = V_{m+1} V_n + \sum_{i=1}^{n-1} \sum_{j=1}^{m} b_{i+m-j}\, V_j\, V_{n-i}.$$

Proof. (4.4) follows from (3.6), (2.3) and Lemma 4.1. (4.5) follows from (2.2) applied to B with $r = k+1$, $s = k+m$, $t = k+m+n-1$, using Lemma 4.1.

Corollary 4.1. If $b_i = 0$ for $i \ge k \ge 1$, then for $n \ge k$,

$$(4.6)\quad y(n) = y_{k-1}\, V_{n-k+2} + \sum_{j=0}^{k-2} y_j \sum_{i=k}^{k+j} b_{i-1-j}\, V_{n+1-j}.$$

If $m \ge 1$, $n \ge 1$, then

$$(4.7)\quad V_{m+n} = V_{m+1}\, V_n + \sum_{i=1}^{p} b_i \sum_{j=q_i}^{i-1} V_{m-j}\, V_{n-i-j}.$$

$$p = \min(k-1, m+n-2),\quad q_i = \max(0, i-m).$$

Proof. (4.7) follows from (2.9) with r,s,t as above.

Corollary 4.1 forms the basis of the Miller-Brown algorithm [MB67]. Doubling formulas for the Bernoulli sequence are given in [Tr66].

Finally, the Bernoulli numbers B_n and the Euler numbers E_n have representations as lower Hessenberg determinants [Mu23], [Mu30]. The results of Section 2 may again be used to derive doubling formulas.

Acknowledgment

The author would like to thank Professor J. F. Traub for suggesting this problem and for his many valuable suggestions.

5. Bibliography

[Eu1764] Euler, L., Specimen Algorithmi Singularis, 1764, Opera Omnia, 1st series, Vol. XV, pp. 33-49.

[Ho70] Householder, A. S., The Numerical Treatment of a Single Nonlinear Equation, 1970, McGraw-Hill, New York.

[Ko72] Kogge, P. M., "Parallel Algorithms for the Efficient Solution of Recurrence Problems," September 1972, Digital Systems Laboratory, Stanford University.

[MB67] Miller, J. C. P. and D. J. S. Brown, "An Algorithm for Evaluation of Remote Terms in a Linear Recurrence Relation," 1967, Computer Journal, Vol. 9, pp. 188-190.

[Mu23] Muir, T., The Theory of Determinants in the Historical Order of Development, Vol. 1-4, 1960, Dover, N. Y. Originally published 1906-1923, MacMillan and Co., London.

[Mu30] _____, Contributions to the History of Determinants, 1900-1920, 1930, Blackie and Son, London.

[St73] Stone, H. S., "An Efficient Parallel Algorithm for the Solution of a Tridiagonal System of Equations," 1973, Journal of the ACM, Vol. 20, pp. 27-38.

[Sy1863a] Sylvester, J. J., "On a Remarkable Modification of Sturm's Theorem," 1853, Collected Works, Vol. 1, pp. 606-619.

[Sy1853b] _____, "On a Fundamental Theorem in the Algorithm of Continued Fractions," 1853, Collected Works, Vol. 1, pp. 641-644.

[Sy1862] _____, "On the Integral of the General Equation in Differences," 1862, Collected Works, Vol. 2, pp. 378-322.

[Tr66] Traub, J. F., "A Class of Globally Convergent Iteration Functions for the Solution of Polynomial Equations," 1966, Mathematics of Computation, Vol. 20, pp. 113-138.