MATHEMATICS OF COMPUTATION
VOLUME 53, NUMBER 187
JULY 1989, PAGES 249-264

A Fast Algorithm for Rational Interpolation Via Orthogonal Polynomials

By Ömer Eğecioğlu* and Çetin K. Koç**

Abstract. A new algorithm for rational interpolation is proposed. Given the data set, the algorithm generates a set of orthogonal polynomials by the classical three-term recurrence relation and then uses Newton interpolation to find the numerator and the denominator polynomials of the rational interpolating function. The number of arithmetic operations of the algorithm to find a particular rational interpolant is $O(N^2)$, where $N+1$ is the number of data points. A variant of this algorithm that avoids Newton interpolation can be used to construct all rational interpolants using only $O(N^2)$ arithmetic operations.

1. Introduction. Let $R(m,n)$ be the set of rational functions of the form $r_{m,n}(x) = p_m(x)/q_n(x)$, where $p_m(x)$ and $q_n(x)$ are polynomials of degree m and n, respectively. If a set of $N+1 = m+n+1$ pairs of points (x_i, f_i) for $0 \le i \le N$ is given, then the *rational interpolation* problem is defined as the task of determining a rational function $r_{m,n}(x) \in R(m,n)$ such that

$$r_{m,n}(x_i) = f_i \quad \text{for } 0 \le i \le N, \tag{1.1}$$

where the x_i's are distinct elements. We also assume that the f_i's are the values of a function $f(x)$ at the nodes x_i for all i.

In the case of polynomial interpolation (i.e., $n = 0$) it is always possible to construct a unique polynomial satisfying (1.1), but this is not true for rational interpolation. For fixed m and n, a rational interpolant that satisfies (1.1) at all points may not exist, even though it may be possible to satisfy (1.1) on a subset of the given point set by means of an $r_{\mu,\nu}(x) \in R(\mu,\nu)$ with $\mu < m$ and $\nu < n$. For this particular pair m and n, the point set becomes a *degenerate configuration.* The points (x_j, f_j) for which

$$r_{\mu,\nu}(x_j) \neq f_j$$

are called the *unattainable* points. Roughly speaking, the existence of an unattainable point (x_j, f_j) for $0 \le j \le N$ means that x_j is a common zero of the numerator and the denominator polynomials. Thus a degenerate point set contains one or more unattainable points.

A direct algorithm for rational interpolation solves a set of linear equations

$$p_m(x_i) - f_i q_n(x_i) = 0, \qquad 0 \le i \le N, \tag{1.2}$$

Received June 3, 1987; revised February 4, 1988.
1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 65D05; Secondary 33A65.
Key words and phrases. Rational interpolation, orthogonal polynomials, Hankel matrices.
*Supported in part by NSF Grant No. DCR-8603722.
**Supported in part by Lawrence Livermore National Laboratory Contract No. LLNL-7526225 while the author was with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, California 93106.

to compute the coefficients of the polynomials $p_m(x)$ and $q_n(x)$, the oldest method being an elegant elimination process due to Jacobi [6, pp. 326–330], [9]. Jacobi's method yields exact determinantal formulae for the coefficients of the denominator and numerator polynomials. A more recent algorithm by Schneider and Werner [10] makes use of the barycentric representation of the rational interpolant. Here a set of linear equations of size n is solved by applying Gaussian elimination to compute the coefficients of $q_n(x)$.

Direct methods can be computationally inefficient because the solution of a general set of linear equations may require as much as $O(n^3)$ arithmetic operations. Jacobi's algorithm, on the other hand, results in a Hankel system of linear equations, which in turn can be solved with $O(n^2)$ arithmetic operations.

We remark that alongside the direct algorithms there exists an array of iterative methods to construct $r_{m,n}(x)$, and the reader is referred to [4] for a review of these algorithms.

In this paper we propose a fast direct algorithm for rational interpolation. Given the data set (x_i, f_i) for $0 \leq i \leq N$, the algorithm first generates a sequence $q_0(x)$, $q_1(x), \ldots, q_n(x)$ of orthogonal polynomials on the discrete set $\{x_0, x_1, \ldots, x_N\}$ with respect to certain weights. The values of these polynomials at the nodes x_i for $0 \leq i \leq N$ are computed by using the classical three-term recurrence relation satisfied by orthogonal polynomials. Once the values of a particular polynomial are known, its coefficients can be computed easily via Newton interpolation. The last orthogonal polynomial $q_n(x)$ generated by this algorithm is the required denominator polynomial of the rational interpolant. Once the denominator polynomial $q_n(x)$ is determined, the construction of the numerator $p_m(x)$ becomes a polynomial interpolation problem. We prove that the number of arithmetic operations of this algorithm to find a particular rational interpolant for given values of m and n is $O(N^2)$.

The coefficients of the orthogonal polynomials can also be found without the application of the Newton interpolation algorithm. This is again achieved by using the three-term recurrence formula. The resulting algorithm requires $O(N^2)$ arithmetic operations to generate the polynomials $q_0(x)$, $q_1(x), \ldots, q_N(x)$. If the same reasoning is applied to the data (x_i, f_i^{-1}) for $0 \leq i \leq N$, the coefficients as well as the values of the numerator polynomials $p_0(x)$, $p_1(x), \ldots, p_N(x)$ can be computed. We show that in this way all rational interpolants $r_{m,n}(x)$ with $m + n = N$ and $0 \leq n \leq N$ can be computed using a total of only $O(N^2)$ arithmetic operations provided that $f_i \neq 0$ for all $0 \leq i \leq N$, and the orthogonal polynomials exist.

The outline of this paper is as follows: In Section 2, we describe the Jacobi algorithm for rational interpolation. Section 3 outlines the use of orthogonal polynomials to compute a particular rational interpolant based on a generalization of Jacobi's approach and Newton interpolation. Fast computation of all rational interpolants that avoids Newton interpolation is described in Section 4. This is followed by examples for each one of these algorithms in Section 5, and conclusions and suggestions for further research in Section 6.

2. Jacobi Rational Interpolation Algorithm. Let $[g(x)]_{0\ldots N}$ denote the Nth divided difference of a function $g(x)$ with respect to the node values x_i for

$0 \le i \le N$. Put

$$w(x) = (x - x_0)(x - x_1) \cdots (x - x_N) \tag{2.1}$$

and

$$w_i = w'(x_i) = \prod_{\substack{j=0 \\ j \neq i}}^{N} (x_i - x_j). \tag{2.2}$$

By direct application of the Lagrange interpolation formula one has

$$[g(x)]_{0 \cdots N} = \sum_{i=0}^{N} \frac{g_i}{w_i} \tag{2.3}$$

where $g_i = g(x_i)$. Note that (2.3) gives the leading coefficient of the polynomial of degree at most N that takes the values g_i at the node points x_i for $0 \le i \le N$. Clearly, if $g(x)$ is a polynomial of degree strictly less than N, then $[g(x)]_{0 \cdots N}$ vanishes.

From (1.2) we can write for any $j \ge 0$

$$\frac{x_i^j p_m(x_i)}{w_i} = \frac{x_i^j f_i q_n(x_i)}{w_i} \quad \text{for } 0 \le i \le N, \tag{2.4}$$

and by summing these terms over i we obtain

$$\sum_{i=0}^{N} \frac{x_i^j p_m(x_i)}{w_i} = \sum_{i=0}^{N} \frac{x_i^j f_i q_n(x_i)}{w_i}.$$

Using the definition (2.3) this can be simplified as

$$[x^j p_m(x)]_{0 \cdots N} = [x^j f(x) q_n(x)]_{0 \cdots N}. \tag{2.5}$$

The left-hand side of (2.5) is equal to the Nth divided difference of the polynomial $x^j p_m(x)$. Thus, if j is in the range $0 \le j \le n - 1$, we have

$$\deg(x^j p_m(x)) = j + m \le n - 1 + m = N - 1.$$

Hence,

$$[x^j p_m(x)]_{0 \cdots N} = 0 \quad \text{for } 0 \le j \le n - 1 \tag{2.6}$$

and therefore

$$[x^j f(x) q_n(x)]_{0 \cdots N} = 0 \quad \text{for } 0 \le j \le n - 1. \tag{2.7}$$

The equation (2.7) allows us to compute the coefficients of $q_n(x)$ by solving a set of linear equations. Since there are n equations and $n + 1$ unknowns, one of the parameters can be chosen arbitrarily.

If $q_n(x)$ is represented in the standard power basis,

$$q_n(x) = \sum_{k=0}^{n} b_k x^k,$$

then from (2.7) the b_k's satisfy the following linear system of equations

$$\sum_{i=0}^{N} \frac{x_i^j f_i q_n(x_i)}{w_i} = \sum_{i=0}^{N} \frac{x_i^j f_i}{w_i} \sum_{k=0}^{n} b_k x_i^k = 0, \qquad 0 \le j \le n - 1,$$

which can be put in the form

$$\sum_{k=0}^{n} h_{j+k} b_k = 0, \qquad 0 \le j \le n-1, \tag{2.8}$$

where

$$h_s = \sum_{i=0}^{N} \frac{f_i x_i^s}{w_i}, \qquad 0 \le s \le 2n-1. \tag{2.9}$$

The system of equations given in (2.8) is a Hankel system. As indicated before, we have one degree of freedom in choosing the coefficients of $q_n(x)$ in any rational interpolation. From now on we will assume that $q_n(x)$ is a *monic* polynomial of degree n, that is $b_n = 1$. With this choice, (2.8) can be rewritten as the matrix equation

$$\begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{n-1} \\ h_1 & h_2 & h_3 & \cdots & h_n \\ h_2 & h_3 & h_4 & \cdots & h_{n+1} \\ \vdots & \vdots & \vdots & & \vdots \\ h_{n-1} & h_n & h_{n+1} & \cdots & h_{2n-2} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix} = - \begin{bmatrix} h_n \\ h_{n+1} \\ h_{n+2} \\ \vdots \\ h_{2n-1} \end{bmatrix}. \tag{2.10}$$

Therefore,

$$q_n(x) = \frac{1}{\det(H_{n-1})} \det \begin{bmatrix} h_0 & h_1 & \cdots & h_{n-1} & h_n \\ h_1 & h_2 & \cdots & h_n & h_{n+1} \\ \vdots & \vdots & & \vdots & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} & h_{2n-1} \\ 1 & x & \cdots & x^{n-1} & x^n \end{bmatrix}, \tag{2.11}$$

where

$$H_j = \begin{bmatrix} h_0 & h_1 & \cdots & h_j \\ h_1 & h_2 & \cdots & h_{j+1} \\ \vdots & \vdots & & \vdots \\ h_j & h_{j+1} & \cdots & h_{2j} \end{bmatrix}.$$

As it stands, the determinantal formula (2.11) is computationally inefficient to construct the denominator polynomial $q_n(x)$. However, Trench [12] has provided a fast algorithm to solve the Hankel system (2.10) directly. If all of the matrices H_0, $H_1, \ldots, H_{n-1}$ are nonsingular, then the number of arithmetic operations to invert H_{n-1} is proportional to n^2. Thus the system in (2.10) can be solved in $O(n^2)$ arithmetic operations given the nonsingularity condition on the principle minors of H_{n-1}.

Note that the existence of the denominator of degree n for $r_{m,n}(x)$ depends only on the nonsingularity of the matrix H_{n-1} by (2.11). The nonsingularity of all the matrices H_0, $H_1, \ldots, H_{n-1}$ is thus equivalent to the existence of all of the denominator polynomials $q_0(x)$, $q_1(x), \ldots, q_n(x)$. It turns out that this is precisely the assumption required on the matrices H_0, $H_1, \ldots, H_{n-1}$ for the construction of orthogonal polynomials outlined in the next section.

Once the coefficients of $q_n(x)$ are found, we can evaluate $q_n(x)$ at all of the node points to check if the data set is degenerate for the given values of m and n and

thus locate unattainable points. The coefficients of $p_m(x)$ can then be computed by performing a polynomial interpolation for the data set $(x_i, f_i q_n(x_i))$ for $0 \le i \le m$.

To summarize, Jacobi's algorithm for rational interpolation can be carried out in the following manner:

Jacobi Algorithm

Input: (x_i, f_i) for $0 \le i \le N$.

Output: Coefficients of $p_m(x)$ and $q_n(x)$ of the rational interpolant $r_{m,n}(x)$.

Step 1. Compute w_i for $0 \le i \le N$ using (2.2).

Step 2. Compute h_s for $0 \le s \le 2n-1$ using (2.9).

Step 3. Solve the resulting Hankel system by using the Trench algorithm to find the coefficients of $q_n(x)$ in the standard form.

Step 4. Check whether or not the data set is degenerate for the given values of m and n, i.e., check if $q_n(x_i) = 0$ for any $0 \le i \le N$ by evaluating $q_n(x)$ at all x_i and thus locating the unattainable points.

Step 5. Interpolate the data set $(x_i, f_i q_n(x_i))$ for $0 \le i \le m$ using the Newton interpolation algorithm to find $p_m(x)$ in the Newton form.

Thus we have

THEOREM 1. *Given the data set* (x_i, f_i) *for* $0 \le i \le N$, *the Jacobi algorithm computes the coefficients of the rational interpolant* $r_{m,n}(x)$ *which satisfies* (1.1) *using* $O(N^2)$ *arithmetic operations.*

Proof. We will count the number of arithmetic operations at each step of the algorithm.

In Step 1, w_i can be calculated with N subtractions and $N-1$ multiplications for any fixed i. For all $i = 0, 1 \ldots, N$ this clearly takes $O(N^2)$ arithmetic operations. In Step 2, first x^s is computed for all $s = 0, 1, \ldots, 2n-1$ with $2n-2$ multiplications. To compute h_s we perform $N+1$ multiplications, $N+1$ divisions and N additions. For all $s = 0, 1, \ldots, 2n-1$ this takes $O(Nn)$ operations. Thus Step 2 takes $O(N^2)$ arithmetic operations.

The solution of the Hankel system takes $O(n^2)$ arithmetic operations as stated earlier. Step 4 consists of evaluating a polynomial of degree n at $N+1$ points. Hence up to Step 5, the number of operations required is no more than $O(N^2)$. Finally, in Step 5 we apply the Newton interpolation to construct the polynomial $p_m(x)$ which takes $\frac{3}{2}m(m+1) = O(m^2) = O(N^2)$ operations [5]. Thus the number of operations of the Jacobi algorithm add up to $O(N^2)$. □

It should be noted that Step 3 of the Jacobi algorithm can be carried out with $O(n^2)$ arithmetic operations without the assumption that the principal minors of H_{n-1} be nonzero (except $\det(H_{n-1})$) by applying an algorithm of Rissanen to solve Hankel and Toeplitz systems [8]. Nevertheless, if one is interested in constructing all $r_{m,n}(x)$ for $m+n = N$ and $0 \le n \le N$, we see that the construction of all of the denominator polynomials $q_0(x), q_1(x), \ldots, q_n(x)$ would require the solution of n such systems, resulting in an $O(n^3)$ algorithm. Given the nonsingularity assumption on the matrices $H_0, H_1, \ldots, H_{n-1}$, we can compute *all* of the polynomials $q_0(x)$,

$q_1(x), \ldots, q_n(x)$ with a total of $O(n^2)$ arithmetic operations by using orthogonal polynomials. This is the subject matter of the next section.

Alternatively, the process that yielded the coefficients of $q_n(x)$ can be repeated with the data set (x_i, f_i^{-1}) for $0 \le i \le N$ to construct the coefficients of $p_m(x)$. To this end, note that (1.2) can be written as

$$q_n(x_i) - \frac{1}{f_i} p_m(x_i) = 0, \qquad 0 \le i \le N,$$

provided $f_i \neq 0$ for all $0 \le i \le N$. By representing $p_m(x)$ in its standard form

$$p_m(x) = \sum_{k=0}^{m} a_k x^k$$

and applying the same reasoning as in the Eqs. (2.4) through (2.8), we obtain

$$\sum_{k=0}^{m} h'_{j+k} a_k = 0, \quad 0 \le j \le m-1, \tag{2.12}$$

where

$$h'_s = \sum_{i=0}^{N} \frac{x_i^s}{f_i w_i}, \qquad 0 \le s \le 2m-1. \tag{2.13}$$

Again the system of equations in (2.12) becomes a Hankel system. By assuming $p_m(x)$ to be a monic polynomial for the moment, we obtain a system of linear equations of size m with the unknowns $a_0, a_1, \ldots, a_{m-1}$ similar to (2.10):

$$\begin{bmatrix} h'_0 & h'_1 & h'_2 & \cdots & h'_{m-1} \\ h'_1 & h'_2 & h'_3 & \cdots & h'_m \\ h'_2 & h'_3 & h'_4 & \cdots & h'_{m+1} \\ \vdots & \vdots & \vdots & & \vdots \\ h'_{m-1} & h'_m & h'_{m+1} & \cdots & h'_{2m-2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{m-1} \end{bmatrix} = \begin{bmatrix} h'_m \\ h'_{m+1} \\ h'_{m+2} \\ \vdots \\ h'_{2m-1} \end{bmatrix}. \tag{2.14}$$

Since $p_m(x)$ and $q_n(x)$ cannot be forced to be monic at the same time, we need to multiply one of these polynomials with a nonzero constant to make them consistent with (1.1). Thus if $q_n(x)$ and $p_m(x)$ are monic polynomials as solutions of the equations (2.10) and (2.14), the corresponding rational interpolant is simply

$$r_{m,n}(x) = \frac{a_m p_m(x)}{q_n(x)}, \tag{2.15}$$

where

$$a_m = \frac{f_0 q_n(x_0)}{p_m(x_0)},$$

assuming that (x_0, f_0) is an attainable point. If we put

$$H'_j = \begin{bmatrix} h'_0 & h'_1 & \cdots & h'_j \\ h'_1 & h'_2 & \cdots & h'_{j+1} \\ \vdots & \vdots & & \vdots \\ h'_j & h'_{j+1} & \cdots & h'_{2j} \end{bmatrix} \tag{2.16}$$

and use the Trench algorithm to solve (2.14), we see that the matrices $H'_0, H'_1, \ldots, H'_{m-1}$ should be nonsingular.

Thus, Newton interpolation can be avoided in Step 5 of the Jacobi Algorithm by solving another Hankel system to determine $p_m(x)$. Since the number of arithmetic operations to solve a Hankel system or to perform a Newton interpolation is the same, $O(m^2)$, we see that this approach does not provide any gains in terms of number of arithmetic operations. However, we will make use of this idea in Section 4 to devise an algorithm to construct all rational interpolants using only $O(N^2)$ arithmetic operations.

The Jacobi algorithm may have undesirable numerical properties because of the stability issues involved in the way the divided differences (i.e., the individual h_s's) are calculated. The algorithm proposed by Schneider and Werner in [10] makes use of the barycentric representation of the rational interpolant and seems to have better numerical properties. On the other hand, there is a trade-off in terms of time complexity, since the linear system of equations that arise in this particular algorithm has no additional structure to facilitate a reduction in the overall time complexity. In particular, the use of Gaussian elimination to solve a general system of linear equations, as proposed, requires $O(n^3)$ operations. Thus the number of arithmetic operations required by the Schneider-Werner algorithm to compute a particular rational interpolant is $O(N^3)$ because of this apparent bottleneck, as opposed to $O(N^2)$ achieved by the Jacobi algorithm.

3. Rational Interpolation Using Orthogonal Polynomials. Our point of departure will be the Jacobi algorithm and we will show that it is possible to compute the denominator polynomial $q_n(x)$ without solving a system of linear equations.

We start by noting that any polynomial of degree j in x_i can be used as a multiplier in Eq. (2.4). It follows that (2.5) and (2.7) hold for an arbitrary polynomial $t_j(x)$ degree j in place of x^j. Hence we have

$$[t_j(x)f(x)q_n(x)]_{0\cdots N} = \sum_{i=0}^{N} \frac{f_i}{w_i} t_j(x_i)q_n(x_i) = 0 \tag{3.1}$$

for $0 \le j \le n-1$. Set $c_i = f_i/w_i$ for $0 \le i \le N$ and define a discrete symmetric bilinear form $\langle \cdot, \cdot \rangle$ on the space of polynomials of degree less than or equal to n by setting

$$\langle t_j(x), t_k(x) \rangle := \sum_{i=0}^{N} c_i t_j(x_i) t_k(x_i). \tag{3.2}$$

Using (3.2), the linear system (3.1) can be written as

$$\sum_{i=0}^{N} c_i t_j(x_i) q_n(x_i) = \langle t_j(x), q_n(x) \rangle = 0, \qquad 0 \le j \le n-1. \tag{3.3}$$

Note that the bilinear form (3.2) does not necessarily define an inner product since it is possible to have $\langle t_j(x), t_j(x) \rangle \le 0$. For our purposes, however, it suffices to assume nondegeneracy, that is, $\langle t_j(x), t_j(x) \rangle \ne 0$ for $0 \le j \le n$. If we require the set $\{t_0(x), t_1(x), \ldots, t_n(x)\}$ to be orthogonal with respect to this discrete bilinear form, then the nondegeneracy condition together with (3.3) imply that $q_j(x)$ is a constant multiple of $t_j(x)$ for $j = 0, 1, \ldots, n$. In particular, the denominator polynomials $q_0(x)$, $q_1(x), \ldots, q_n(x)$ are orthogonal with respect to (3.2).

The nondegeneracy of the bilinear form (3.2) is in turn guaranteed by the nonsingularity of the matrices $H_0, H_1, \ldots, H_{n-1}$.

THEOREM 2. *There exists a sequence of orthogonal polynomials* $\{t_0(x), t_1(x), \ldots, t_n(x)\}$ *on the set* $\{x_0, x_1, \ldots, x_N\}$ *with respect to the symmetric bilinear form*

$$\langle t_j(x), t_k(x)\rangle = \sum_{i=0}^{N} \frac{f_i}{w_i} t_j(x_i) t_k(x_i)$$

if and only if the matrices $H_0, H_1, \ldots, H_{n-1}$ *are all nonsingular.*

Proof. From the general theory of orthogonal polynomials [1], the polynomial $t_j(x)$ is given explicitly (up to a constant multiple) by the determinant

$$t_j(x) = \det \begin{bmatrix} \langle 1,1\rangle & \langle 1,x\rangle & \cdots & \langle 1,x^j\rangle \\ \langle x,1\rangle & \langle x,x\rangle & \cdots & \langle x,x^j\rangle \\ \vdots & \vdots & & \vdots \\ \langle x^{j-1},1\rangle & \langle x^{j-1},x\rangle & \cdots & \langle x^{j-1},x^j\rangle \\ 1 & x & \cdots & x^j \end{bmatrix}. \tag{3.4}$$

Note that

$$\langle x^k, x^l\rangle = \sum_{i=0}^{N} \frac{f_i}{w_i} x_i^{k+l} = h_{k+l}.$$

Thus the coefficient of x^j in $t_j(x)$ is precisely the determinant of H_{j-1}, which has to be nonzero. □

Since we have assumed that $q_n(x)$ is monic, by making use of (3.4) we arrive at Jacobi's explicit formula for the denominator polynomials given in (2.11).

The conventional method for generating orthogonal polynomials is the Gram-Schmidt orthogonalization process. However in our case, this approach would require $O(n^2)$ inner product operations to generate $q_n(x)$. A more efficient technique is to use the three-term recurrence relation for orthogonal polynomials to generate the values of the polynomials $q_j(x)$ directly. With this approach, the total number of inner product operations required to compute the values $q_0(x_i), q_1(x_i), \ldots, q_{n-1}(x_i)$ for any given point x_i becomes only $O(n)$ [3], [7], [11].

More precisely, let $\{t_0(x), t_1(x), \ldots, t_n(x)\}$ be a set of polynomials satisfying the orthogonality relationship with respect to the weights c_i and the sequence of data points x_i for $0 \le i \le N$. It is well known that

$$\begin{aligned} t_{-1}(x) &= 0, \qquad t_0(x) = 1, \\ t_{j+1}(x) &= (x - \alpha_j) t_j(x) - \beta_j t_{j-1}(x), \qquad j = 0, 1, \ldots, \end{aligned} \tag{3.5}$$

where α_j and β_j are constants determined as

$$\alpha_j = \frac{\langle x t_j(x), t_j(x)\rangle}{\langle t_j(x), t_j(x)\rangle}, \qquad \beta_j = \frac{\langle t_j(x), t_j(x)\rangle}{\langle t_{j-1}(x), t_{j-1}(x)\rangle}. \tag{3.6}$$

Note that the relation (3.5) generates monic orthogonal polynomials, and requires only the nondegeneracy of the underlying symmetric bilinear form.

Let $T = [T_{ji}]$ be the $(N+1) \times (N+1)$ matrix where

$$T_{ji} = t_j(x_i), \qquad 0 \le i, j \le N. \tag{3.7}$$

In other words, the jth row of T consists of the vector of values of the jth orthogonal polynomial $t_j(x)$ at the nodes $x_0, x_1, \ldots, x_N$. The following procedure generates the first $n+1$ rows of the matrix T using the three-term recursion (TTR) in (3.5).

Procedure TTR.
Input: n and (x_i, f_i) for $0 \le i \le N$.
Output: $T = [T_{ji}]$ for $0 \le j \le n$ and $0 \le i \le N$.
Step 1. Compute w_i and c_i for $0 \le i \le N$ using

$$w_i = \prod_{\substack{j=0 \\ j \ne i}}^{N} (x_i - x_j) \quad \text{and} \quad c_i = \frac{f_i}{w_i}.$$

Step 2. Set $T_{0i} = 1$ for $0 \le i \le N$, $\beta_0 = 0$, and compute

$$\gamma_0 = \sum_{i=0}^{N} c_i x_i, \quad \theta_0 = \sum_{i=0}^{N} c_i, \quad \text{and} \quad \alpha_0 = \frac{\gamma_0}{\theta_0}.$$

Step 3. Set $T_{1i} = x_i - \alpha_0$ for $0 \le i \le N$, and compute

$$\gamma_1 = \sum_{i=0}^{N} c_i x_i T_{1i}^2, \quad \theta_1 = \sum_{i=0}^{N} c_i T_{1i}^2, \quad \text{and} \quad \alpha_1 = \frac{\gamma_1}{\theta_1}, \quad \beta_1 = \frac{\theta_1}{\theta_0}.$$

Step 4. For $1 \le j \le n-1$ compute

$$T_{j+1,i} = (x_i - \alpha_j) T_{ji} - \beta_j T_{j-1,i}, \qquad 0 \le i \le N,$$
$$\gamma_{j+1} = \sum_{i=0}^{N} c_i x_i T_{j+1,i}^2, \qquad \theta_{j+1} = \sum_{i=0}^{N} c_i T_{j+1,i}^2,$$
$$\alpha_{j+1} = \frac{\gamma_{j+1}}{\theta_{j+1}}, \qquad \beta_{j+1} = \frac{\theta_{j+1}}{\theta_j}.$$

We will denote by $T := \mathrm{TTR}(n, x_i, f_i)$ the $(n+1) \times (N+1)$ matrix produced by TTR from the input data n and (x_i, f_i), $0 \le i \le N$.

LEMMA 1. *Given the data set (x_i, f_i) for $0 \le i \le N$, and $n \le N$, Procedure TTR computes the first $n+1$ rows of the matrix T using*

$$2N^2 + 10Nn + 2N + 10n - 1 = O(N^2) \tag{3.8}$$

arithmetic operations. In particular, all rows of T can be computed with

$$12N^2 + 12N - 1 = O(N^2) \tag{3.9}$$

arithmetic operations.

Proof. In Step 1 the computation of w_i takes $2N-1$ operations for a particular value of i. All $0 \le i \le N$ takes $(N+1)(2N-1)$ operations. The c_i's can be computed in $N+1$ operations. Hence Step 1 of the procedure takes $2N^2+2N$ arithmetic operations. The computation of α_0 in Step 2 takes $3N+2$ arithmetic operations. In Step 3, T_{1i} and T_{1i}^2 are computed with $2(N+1)$ arithmetic operations. Then to compute α_1 and β_1 we need to perform $5N+5$ arithmetic operations. Thus Step 3 takes $7N+7$ operations. In Step 4 for a particular value of j the computation of $T_{j+1,i}$ and $T_{j+1,i}^2$ takes $5N+5$ arithmetic operations. Then to compute α_{j+1} and β_{j+1}, $5N+5$ operations are needed. For all $1 \le j \le n-1$, Step 4 takes $(n-1)(10N+10)$ operations. The result in (3.8) thus follows. To find (3.9) we replace n with N in (3.8). □

We are now in a position to describe the rational interpolation algorithm via orthogonal polynomials and to prove that rational interpolation can be done using $O(N^2)$ arithmetic operations. The following algorithm first generates the first $n+1$ rows of the matrix $Q = [Q_{ji}]$, where $Q_{ji} = q_j(x_i)$, and then applies the Newton interpolation algorithm to find $q_n(x)$ and $p_m(x)$.

Algorithm 1.

Input: (x_i, f_i) for $0 \le i \le N$.

Output: Coefficients of $p_m(x)$ and $q_n(x)$ of the rational interpolant $r_{m,n}(x)$.

Step 1. $Q := \mathrm{TTR}(n, x_i, f_i)$.

Step 2. Interpolate the data set (x_i, Q_{ni}) for $0 \le i \le n$ to compute the coefficients of $q_n(x)$ in the Newton form by using the Newton interpolation algorithm.

Step 3. Compute $f_i Q_{ni}$ for all $0 \le i \le N$ and then interpolate the data set $(x_i, f_i Q_{ni})$ for $0 \le i \le m$ to compute the coefficients of $p_m(x)$ in the Newton form by using the Newton interpolation algorithm.

THEOREM 3. *Given the data set (x_i, f_i) for $0 \le i \le N$, Algorithm 1 computes the coefficients of the rational interpolant $r_{m,n}(x)$ which satisfies (1.1) using $O(N^2)$ arithmetic operations.*

Proof. By Lemma 1, Step 1 takes $2N^2+10Nn+2N+10n-1$ arithmetic operations. Step 2 is an interpolation process with $n+1$ points. This requires $\frac{3}{2}n(n+1)$ operations [5]. Similarly, in Step 3 we first compute $f_i Q_{ni}$ for all $0 \le i \le m$ and then perform a polynomial interpolation with $m+1$ points; hence, $m+1+\frac{3}{2}m(m+1)$ arithmetic operations need to be performed in this step. If we sum the number of operations at each step we conclude that Algorithm 1 takes

$$2N^2 + \tfrac{3}{2}(n^2+m^2) + 10Nn + 2N + \tfrac{23}{2}n + \tfrac{5}{2}m = O(N^2) \tag{3.10}$$

arithmetic operations. □

One remarkable property of Algorithm 1 is that it makes it trivial to check whether the given data set is degenerate for any values of m and n with $m+n=N$. To check for degeneracy, we compute the values of Q_{ji} for all $0 \le j \le N$ in Step 1. This implies that the input set for the Procedure TTR is N and (x_i, f_i) for $0 \le i \le N$. The number of arithmetic operations to compute all rows Q is $12N^2+12N-1$ as given in (3.9). Thus the total number of arithmetic operations

of Algorithm 1 increases slightly to

$$12N^2 + \tfrac{3}{2}(n^2 + m^2) + 12N + \tfrac{3}{2}n + \tfrac{5}{2}m, \tag{3.11}$$

which is still $O(N^2)$. This allows us to select a particular value of m and n for which the data set is not degenerate while keeping the total number of operations within $O(N^2)$.

4. Fast Computation of All Rational Interpolants. Note that it is possible to compute the coefficients of the polynomial $q_n(x)$ recursively using the three-term recurrence formula in (3.5) and thus avoid the Newton interpolation in Step 2 of Algorithm 1. This can be done by applying the three-term recursion directly to the coefficients of the denominator polynomials. More precisely, let $B = [B_{jk}]$ be the $(N+1) \times (N+1)$ matrix in which the jth row consists of the coefficients of the polynomial $q_j(x)$, i.e.,

$$q_j(x) = \sum_{k=0}^{j} B_{jk} x^k. \tag{4.1}$$

Then B is a lower triangular matrix with unit diagonal whose elements satisfy the recursion

$$B_{j+1,k} = B_{j,k-1} - \alpha_j B_{jk} - \beta_j B_{j-1,k}, \qquad 0 \le k < j \le N-1, \tag{4.2}$$

induced by (3.5). In (4.2) we take

$$\begin{aligned} B_{j,-1} &= 0 \quad \text{for } 0 \le j \le N, \\ B_{-1,k} &= 0 \quad \text{for } 0 \le k \le N. \end{aligned} \tag{4.3}$$

Thus using this recursion formula we can generate the coefficients of the polynomials $q_j(x)$ for $0 \le j \le n$. The values of the polynomials $q_j(x)$ at the node points x_i (i.e., Q_{ji}) are computed at each step to calculate α_j and β_j, but this also helps to locate the unattainable points if the point set happens to be degenerate for the particular values of m and n.

If only one rational interpolant is needed then, in Step 2 of Algorithm 1, the choice between the Newton interpolation algorithm or the application of recurrence formula in (4.2) is somewhat arbitrary, since both algorithms will require $O(n^2)$ arithmetic operations. For the Newton algorithm, however, the constant in front of the order is smaller.

More importantly, the generation of the coefficients of the polynomials $q_j(x)$, $0 \le j \le n$, in $O(n^2)$ arithmetic operations suggests a drastic cut-down on the number of arithmetic operations when all rational interpolants $r_{m,n}(x)$ for $0 \le n \le N$ with $m = N - n$ are computed. We note that for $0 \le j \le N$ the coefficients of the polynomials $p_j(x)$ can also be computed similarly by applying the recursion in (4.2) to the data (x_i, f_i^{-1}) for $0 \le i \le N$, as we already remarked at the end of Section 2. This is possible provided $f_i \ne 0$ for $0 \le i \le N$ and $H'_0, H'_1, \ldots, H'_{N-1}$ are nonsingular. This given, define $A = [A_{jk}]$ and $P = [P_{ij}]$ to be $(N+1) \times (N+1)$ matrices where the jth row of A contains the coefficients of the polynomial $p_j(x)$,

$$p_j(x) = \sum_{k=0}^{j} A_{jk} x^k, \tag{4.4}$$

and the jth row of P contains the values of the polynomial $p_j(x)$ at the nodes x_i

$$P_{ji} = p_j(x_i), \qquad 0 \le i, j \le N,$$

similar to the matrices B and Q.

The following algorithm first generates the values and the coefficients of the polynomials $q_j(x)$ for all $0 \le j \le N$. The algorithm then proceeds to generate the values and the coefficients of the polynomials $p_j(x)$ for $0 \le j \le N$ by applying the same technique to the data (x_i, f_i^{-1}) for $0 \le i \le N$.

Algorithm 2.

Input: (x_i, f_i) for $0 \le i \le N$.

Output: Coefficients of $p_m(x)$ and $q_n(x)$ for $0 \le n \le N$ and $m = N - n$.

Step 1. $Q := \mathrm{TTR}(N, x_i, f_i)$.

Step 2. Set $B_{jj} = 1$ for $0 \le j \le N$ and $B_{jk} = 0$ for $0 \le j < k \le N$. Set $B_{10} = -\alpha_0$. For all $0 \le k < j \le N - 1$ compute

$$B_{j+1,k} = B_{j,k-1} - \alpha_j B_{jk} - \beta_j B_{j-1,k}.$$

Step 3. Compute f_i^{-1} for $0 \le i \le N$ and $P := \mathrm{TTR}(N, x_i, f_i^{-1})$.

Step 4. Set $A_{jj} = 1$ for $0 \le j \le N$ and $A_{jk} = 0$ for $0 \le j < k \le N$. Set $A_{10} = -\alpha_0$. For all $0 \le k < j \le N - 1$ compute

$$A_{j+1,k} = A_{j,k-1} - \alpha_j A_{jk} - \beta_j A_{j-1,k}.$$

Step 5. Update the coefficients of $p_j(x)$ according to (2.15):

$$A_{jk} = f_0 \frac{Q_{N-j,0}}{P_{j0}} A_{jk}, \qquad 0 \le k \le j \le N.$$

At the end of Algorithm 2, the coefficients of the polynomials $p_m(x)$ and $q_n(x)$ are A_{mk} and B_{nk}, respectively, for $0 \le k \le N$. Note that $A_{mk} = 0$ for $k > m$ and $B_{nk} = 0$ for $k > n$.

THEOREM 4. *Algorithm* 2 *computes all rational interpolants* $r_{m,n}(x)$ *for* $n + m = N$ *and* $n = 0, 1, \ldots, N$ *using* $O(N^2)$ *arithmetic operations provided the matrices* $H_0, H_1, \ldots, H_{N-1}$ *and* $H_0', H_1', \ldots, H_{N-1}'$ *are nonsingular.*

Proof. As we showed in Theorem 2, the orthogonal polynomials $q_0(x), q_1(x), \ldots, q_N(x)$ can be constructed if and only if the matrices $H_0, H_1, \ldots, H_{N-1}$ are all nonsingular. By applying the same technique to the data (x_i, f_i^{-1}) for $0 \le i \le N$, we conclude that the polynomials $p_0(x), p_1(x), \ldots, p_N(x)$ can be constructed if the matrices $H_0', H_1', \ldots, H_{N-1}'$ are all nonsingular and if $f_i \ne 0$ for all $0 \le i \le N$.

To compute the number of arithmetic operations required for Algorithm 2 we count the operations at each step. Step 1 takes $12N^2 + 12N - 1$ arithmetic operations by Lemma 1. Also Step 3 will take $N + 1 + 12N^2 + 12N - 1$ arithmetic operations. For each of Step 2 and 4 notice that the recursion in (4.2) requires $\sum_{j=1}^{N-1} \sum_{k=0}^{j} 4 = 2N^2 + 2N - 4$ arithmetic operations. Also in Step 5 only the lower triangular part of the matrix A is updated which results in

$$2N + 2 + \sum_{j=1}^{N-1} \sum_{k=0}^{j} 1 = \frac{1}{2}N^2 + \frac{7}{2}N + 3$$

arithmetic operations. Thus Algorithm 2 requires a total of

$$\frac{53}{2}N^2 + \frac{61}{2}N - 6 = O(N^2) \tag{4.5}$$

arithmetic operations to compute all rational interpolants. □

5. Examples. The algorithms have been implemented on a VAX-11/780 computer running Unix 4.2 BSD, using the *Pascal* programming language. Even though we have not yet conducted a detailed experimental study of the numerical properties of these algorithms, we present computer generated solutions to two simple interpolation problems.

Example 1 (Algorithm 1). Given $f(x) = |x|$, find $r_{2,2}(x)$ which interpolates $f(x)$ at the node points -1, -0.5, 0, 0.5, 1. We apply Step 1 of Algorithm 1 to the data and obtain the following Q matrix of size (5×5):

$$Q = \begin{bmatrix} 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ -1.00 & -0.50 & 0.00 & 0.50 & 1.00 \\ 1.50 & 0.75 & 0.50 & 0.75 & 0.75 \\ 0.75 & 0.75 & 0.00 & -0.75 & -0.75 \\ 0.00 & 0.00 & 0.25 & 0.00 & 0.00 \end{bmatrix}.$$

The rows of the matrix Q are the values of the denominator polynomials $q_j(x)$ at the nodes x_i for $0 \le i, j \le 4$. As mentioned before, if $Q_{ni} = 0$ for an i then the point (x_i, f_i) is an unattainable point for the (m, n) pair. An inspection of all entries in Q gives the following result:

m	n	unattainable points
4	0	none
3	1	(0,0)
2	2	none
1	3	(0,0)
0	4	all points except (0,0)

The rational interpolant $r_{2,2}(x)$ interpolates $f(x)$ at all points. In order to determine $q_2(x)$, we use the Newton interpolation algorithm. The values of the polynomial $q_2(x)$ at the node points are seen from the third row to be

$$q_2(-1) = 1.50, \quad q_2(-0.5) = 0.75, \quad q_2(0) = 0.50, \quad q_2(0.5) = 0.75, \quad q_2(1) = 0.75.$$

Since three points are necessary and sufficient to determine the coefficients of $q_2(x)$, we choose the first three points: $(-1,\ 1.50)$ $(-0.5,\ 0.75)$, $(0,0,\ 0.5)$, and find $q_2(x) = x^2 + 0.5$. In order to determine the coefficients of the numerator polynomial, first we compute

$$f_0 Q_{20} = 1.50, \quad f_1 Q_{21} = 0.375, \quad f_2 Q_{22} = 0.00$$

and then apply Newton interpolation to the data $(-1,\ 1.50)$, $(-0.5,\ 0.375)$, $(0,\ 0.00)$ to find the numerator polynomial $p_2(x) = 1.50x^2$. Thus,

$$r_{2,2}(x) = \frac{p_2(x)}{q_2(x)} = \frac{1.50x^2}{x^2 + 0.5}.$$

Here we note that since $f_2 = 0$, Algorithm 2 cannot be applied to find all rational interpolants for this problem.

Example 2 (Algorithm 2). Given $f(x) = 2^x$, find all rational interpolants of $f(x)$ at the node points $-2, -1, 0, 1, 2$. The application of Step 1 and Step 2 of Algorithm 2 to this data set produces the following Q and B matrices:

$$Q = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -8 & -7 & -6 & -5 & -4 \\ 48 & 36 & 26 & 18 & 12 \\ -192 & -120 & -72 & -42 & -24 \\ 384 & 192 & 96 & 48 & 24 \end{bmatrix},$$

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -6 & 1 & 0 & 0 & 0 \\ 26 & -9 & 1 & 0 & 0 \\ -72 & 38 & -9 & 1 & 0 \\ 96 & -66 & 23 & -6 & 1 \end{bmatrix}.$$

Thus we observe that for this interpolation problem all points are attainable for all (m, n) pairs. The elements of B are the coefficients of the denominator polynomials $q_j(x)$ for $0 \le j \le 4$. In other words, the entries in the first column are the coefficient of x^0 in $q_j(x)$, the entries in the next column are the coefficient of x, and so on. The unit diagonal entries correspond to the coefficient of the leading term x^j in $q_j(x)$.

Similarly, the application of Steps 3, 4 and 5 of Algorithm 2 produces the coefficients of the numerator polynomials, namely the A matrix:

$$A = \begin{bmatrix} 96 & 0 & 0 & 0 & 0 \\ -72 & -12 & 0 & 0 & 0 \\ 26 & 9 & 1 & 0 & 0 \\ -6 & -\frac{19}{6} & -\frac{3}{4} & -\frac{1}{12} & 0 \\ 1 & \frac{11}{16} & \frac{23}{96} & \frac{1}{16} & \frac{1}{96} \end{bmatrix}.$$

Hence the rational interpolants are found to be

$$r_{0,4}(x) = \frac{p_0(x)}{q_4(x)} = \frac{96}{x^4 - 6x^3 + 23x^2 - 66x + 96},$$

$$r_{1,3}(x) = \frac{p_1(x)}{q_3(x)} = \frac{12x + 72}{-x^3 + 9x^2 - 38x + 72},$$

$$r_{2,2}(x) = \frac{p_2(x)}{q_2(x)} = \frac{x^2 + 9x + 26}{x^2 - 9x + 26},$$

$$r_{3,1}(x) = \frac{p_3(x)}{q_1(x)} = \frac{x^3 + 9x^2 + 38x + 72}{-12x + 72},$$

$$r_{4,0}(x) = \frac{p_4(x)}{q_0(x)} = \frac{x^4 + 6x^3 + 23x^2 + 66x + 96}{96}.$$

6. Summary and Conclusions. We have described two algorithms for rational interpolation. Given that the Hankel matrices $H_0, H_1, \ldots, H_{N-1}$ are all nonsingular, the first one of these algorithms (Algorithm 1) generates orthogonal polynomials $q_0(x), q_1(x), \ldots, q_N(x)$ which are the monic denominator polynomials

of the associated rational interpolants. If the computation of the corresponding numerator polynomials are then carried out by Newton interpolation, then the number of arithmetic operations required becomes $O(N^3)$. However, if the corresponding matrices $H'_0, H'_1, \ldots, H'_{N-1}$ for the numerators are also nonsingular, then Newton interpolation can be avoided, and all rational interpolants can be found with only $O(N^2)$ arithmetic operations. This is the content of Algorithm 2.

Note that in case a particular matrix H_{j-1} happens to be singular, then it is no longer possible to continue the generation of the denominator polynomials $q_j(x)$, $q_{j+1}(x), \ldots, q_N(x)$ by using the three-term recursion, as the necessary condition for the existence of the desired orthogonal family of polynomials is not satisfied. Thus, if the discrete symmetric bilinear form we are interested in happens to be degenerate, it is not clear how to proceed with the method of orthogonal polynomials to generate the rest of the rational interpolants. At this point, however, the remaining denominator polynomials can be computed individually by repeated application of Rissanen's algorithm at a cost of $O(N^2)$ operations each. Similarly, the singularity of a matrix H'_{j-1} associated with a numerator polynomial puts a limit on the applicability and the operating range of Algorithm 2. Nevertheless, it is possible to combine these two approaches for a hybrid algorithm whose running time is guaranteed to be no worse than the existing algorithms for rational interpolation. In addition, both Algorithm 1 and Algorithm 2 provide simple provisions to check for degeneracy of the interpolation problem at hand.

Although the proposed algorithms are fast, an extensive analysis of their practicality in terms of numerical stability has not yet been carried out. It should be noted however, that both Algorithm 1 and Algorithm 2 can be parallelized on either shared-memory or message-passing multiprocessor models. This aspect of the algorithms will be reported elsewhere [2].

Acknowledgment. We are grateful to the anonymous referee whose comments and suggestions have greatly improved the readability of this paper.

Department of Computer Science
University of California
Santa Barbara, California 93106
E-mail: omer@trurl.ucsb.edu

Department of Electrical Engineering
University of Houston
Houston, Texas 77204
E-mail: cetin@izmir.ee.uh.edu

1. P. J. DAVIS, *Interpolation and Approximation*, Dover, New York, 1975.

2. Ö. EĞECIOĞLU & Ç. K. KOÇ, *An $O(N \log N)$ Parallel Algorithm for Rational Interpolation*, Technical Report No. TRCS88-2, Department of Computer Science, University of California, Santa Barbara, August 1988.

3. G. E. FORSYTHE, "Generation and use of orthogonal polynomials for data fitting with a digital computer," *J. Soc. Indust. Appl. Math.*, v. 5, 1957, pp. 74–78.

4. P. R. GRAVES-MORRIS & T. R. HOPKINS, "Reliable rational interpolation," *Numer. Math.*, v. 36, 1981, pp. 111–128.

5. F. B. HILDEBRAND, *An Introduction to Numerical Analysis*, McGraw-Hill, New York, 1956.

6. T. MUIR, *The Theory of Determinants*, vol. 2, Dover, New York, 1960.

7. A. RALSTON & P. RABINOWITZ, *A First Course in Numerical Analysis*, McGraw-Hill, New York, 1978.

8. J. RISSANEN, "Solution of linear equations with Hankel and Toeplitz matrices," *Numer. Math.*, v. 22, 1974, pp. 361–366.

9. T. J. RIVLIN, *An Introduction to the Approximation of Functions*, Dover, New York, 1969.

10. C. SCHNEIDER & W. WERNER, "Some new aspects of rational interpolation," *Math. Comp.*, v. 47, 1986, pp. 285–299.

11. G. SZEGÖ, *Orthogonal Polynomials*, Amer. Math. Soc., Providence, R. I., 1959.

12. W. F. TRENCH, "An algorithm for the inversion of finite Hankel matrices," *J. Soc. Indust. Appl. Math.*, v. 13, 1965, pp. 1102–1107.