Numer. Math. 64, 115–126 (1993)

Numerische Mathematik

A fast algorithm for scalar Nevanlinna-Pick interpolation*

Çetin Kaya Koç and Guanrong Chen
Department of Electrical Engineering, University of Houston, Houston, TX 77204, USA

Received October 9, 1990 / Revised version received April 29, 1992

Summary. In this paper, we derive a fast algorithm for the scalar Nevanlinna-Pick interpolation. Given n distinct points z_i in the unit disk $|z| < 1$ and n complex numbers w_i satisfying the Pick condition for $1 \leqq i \leqq n$, the new Nevanlinna-Pick interpolation algorithm requires only $O(n)$ arithmetic operations to evaluate the interpolatory rational function at a particular value of z, in contrast to the classical algorithm which requires $O(n^2)$ arithmetic operations to compute the so-called Fenyves array (which is inherent in the classical algorithm). The new algorithm bypasses the generation of the Fenyves array to speed up the computation, and also yields a parallel scheme requiring only $O(\log n)$ arithmetic operations on a concurrent-read, exclusive-write parallel random access machine with n processors. We must remark that the rational function $f(z)$ computed by the new algorithm is one degree higher than the function computed by the classical algorithm.

Mathematics Subject Classification (1991): 65D05, 68Q25, 93B40

1. Introduction

The Nevanlinna-Pick interpolation technique plays an important role not only in the classical interpolation and approximation theories but also in modern systems control theory and signal processing. For further information on the engineering applications of the Nevanlinna-Pick interpolation, the reader is referred to [5] and [2].

The literature on the Nevanlinna-Pick interpolation contains only one main computational scheme, henceforth referred to as the classical Nevanlinna-Pick interpolation algorithm. This algorithm, as well as some computational schemes

* Supported in part by the US Army Research Office Grant No. DAAL03-91-G-0106

Correspondence to: C.K. Koç, Department of Electrical and Computer Engineering, Oregon State University, Corvallis, OR 97331, USA

which are very mild modifications of it, requires $O(n^2)$ arithmetic operations to compute the Fenyves array (which is inherent in the classical algorithm). Moreover, the classical algorithm requires only $O(n)$ arithmetic operations to evaluate the interpolatory rational function at a given point.

In [5], we have proposed a parallel algorithm for the Nevanlinna-Pick interpolation, introduced some parallel schemes suitable for implementation on shared-memory multiprocessors and systolic/wavefront arrays, and studied their computational complexities. Given n processors, the parallel algorithm in [5] requires $O(n)$ parallel arithmetic operations for computing the Fenyves array and $O(\log n)$ parallel arithmetic operations to evaluate the interpolatory rational function at a given point.

In the present paper, based on some advanced mathematical theory and computational techniques, we propose a new algorithm for the Nevanlinna-Pick interpolation which differs from the classical algorithm in that the generation of the Fenyves array can be bypassed and the interpolatory rational function can be evaluated directly at any given point by using a very simple and straightforward recursive formula. As a result, the new Nevanlinna-Pick interpolation algorithm skips the computation of the Fenyves array, requires only $O(n)$ arithmetic operations to evaluate the interpolatory rational function at any given point, and yields a parallel scheme requiring only $O(\log n)$ arithmetic operations on the concurrent-read, exclusive-write (CREW) version of a parallel random access machine (PRAM) [3, 9, 11] with n processors.

2. Statement of the Nevanlinna-Pick interpolation problem

Let H^∞ be the Hardy space of complex-variable functions analytic in the open unit disk $|z| < 1$ and belonging to $L^\infty(|z| = 1)$ on the unit circle. Let H_1^∞ denote the collection of all functions in H^∞ with $\|f\|_{H^\infty} \leqq 1$. For n given distinct points $z_1, z_2, \ldots, z_n$ in the unit disk $|z| < 1$ and n complex numbers $w_1, w_2, \ldots, w_n$ satisfying the so-called Pick condition that the $n \times n$ matrix

$$\left[\frac{1 - w_i \bar{w}_j}{1 - z_i \bar{z}_j}\right]_{1 \leqq i, j \leqq n}$$

is non-negative definite, find a function $f(z)$ in H_1^∞ which satisfies the interpolation constraints:

$$f(z_i) = w_i \quad \text{for } i = 1, 2, \ldots, n .$$

3. Description of the new algorithm

Define two functions

(1) $$a_k(z) := \frac{z - z_k}{1 - \bar{z}_k z}$$

and

(2) $$b_k(z) := \frac{z + w_k}{1 + \bar{w}_k z}$$

for $k = 1, 2, \ldots, n$. Then, our new algorithm for the Nevanlinna-Pick interpolation is described as follows:

The new Nevanlinna-Pick algorithm

Step 1. Set $f_0(z) = 1$.

Step 2. For $k = 1, 2, \ldots, n$, compute $f_k(z)$ recursively by

$$f_k(z) = b_k(a_k(z) \cdot f_{k-1}(a_k(z))) \,. \tag{3}$$

The final result is $f(z) = f_n(z)$.

The final rational function $f(z)$ obtained in this algorithm is in H_1^∞ and satisfies $f(z_i) = w_i$ *for* $i = 1, 2, \ldots, n$, as required.

We remark that the interpolatory rational function $f(z)$ obtained by using the new algorithm is different from the one given by the classical scheme, in the sense that the new rational function $f(z)$ is one degree higher than the classical result. It must be noticed, however, that the Nevanlinna-Pick interpolation problem stated above does not have any requirement on the degree of the resultant interpolatory rational function $f(z)$. It will be seen that by increasing one degree, we gain a much faster computational algorithm. Before demonstrating this significant improvement, we first give a detailed derivation for the new algorithm.

4. Derivation of the new algorithm

For the simplest case $n = 1$, it is clear that the rational function

$$f_1(z) = b_1(a_1(z)) = \frac{\dfrac{z - z_1}{1 - \bar{z}_1 z} + w_1}{1 + \bar{w}_1 \dfrac{z - z_1}{1 - \bar{z}_1 z}}$$

is in H_1^∞ and satisfies the interpolation condition $f_1(z_1) = w_1$. Hence, we only consider the non-trivial case where $n \geqq 2$. To establish the result for the general case, however, we need some advanced mathematics.

Let V be a linear space with dual V', and $\mathscr{L}(V)$ the space of all linear operators from V into itself. For each A in $\mathscr{L}(V)$, let A' be its adjoint, so that $(x', Ax) = (A'x', x)$ for all $x \in V$ and $x' \in V'$, where $(f, g) := f(g)$. As usual, a subspace $D \subseteq V'$ is called an *invariant subspace under* A' if $A'(D) \subseteq V'$. The following so-called Interpolation Theorem (for the Nudel'man problem) can be found in [10]:

Interpolation Theorem. *Let $A \in \mathscr{L}(V)$ and $b, c \in V$ be given. Let $D \subseteq V'$ be an invariant subspace of V' under A' such that*

$$\sum_{j=0}^{\infty} |(x', A^j c)|^2 < \infty$$

for all $x' \in D$. Then the following two statements are equivalent:

(i) *There exists a symbol*

$$h(z) = \sum_{j=0}^{\infty} h_j z^j$$

in H_1^∞ such that

$$(x', b) = \sum_{j=0}^{\infty} h_j (x', A^j c)$$

for all $x' \in D$.

(ii) *For all $x' \in D$,*

$$\sum_{j=0}^{\infty} |(x', A^j b)|^2 \leqq \sum_{j=0}^{\infty} |(x', A^j c)|^2 .$$

For the Nevanlinna-Pick interpolation problem, we may, without loss of generality, consider the case where $|w_i| < 1$ for all $i = 1, 2, \ldots, n$, since by the Pick condition we have

$$[1\ 0 \ldots 0] \left[\frac{1 - w_k \bar{w}_l}{1 - z_k \bar{z}_l} \right]_{n \times n} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \geqq 0$$

or

$$\frac{1 - |w_1|^2}{1 - |z_1|^2} \geqq 0 .$$

If $|w_1| = 1$, then

$$\frac{1 - |w_1|^2}{1 - |z_1|^2} = 0 .$$

However, since the Pick matrix is Hermitian, we have

$$\left| \frac{1 - w_1 \bar{w}_l}{1 - w_1 \bar{z}_l} \right| \leqq \frac{1 - |w_1|^2}{1 - |z_1|^2} \cdot \frac{1 - |w_l|^2}{1 - |z_l|^2} , \quad l = 1, 2, \ldots, n .$$

Hence, $|w_1| = 1$ implies that $w_l = w_1$ for all $l = 1, 2, \ldots, n$. In other words, since the Pick condition is preassumed, if $|w_1| = 1$ then we must have $w_l = w_1$ for all $l = 1, 2, \ldots, n$, so that the constant function $f(z) = w_1$ is an expected result that satisfies both $f(z) \in H_1^\infty$ and $f(z_i) = w_i$, $i = 1, 2, \ldots, n$.

To verify the algorithm for the nontrivial case where $|w_i| < 1$ for all $i = 1, 2, \ldots, n$, we first observe that if $f(z) = f_n(z)$ is an expected result, then we must have

$$w_i = f(z_i) = b_n(a_n(z_i) f_{n-1}(a_n(z_i)))$$

$$= \frac{a_n(z_i) f_{n-1}(a_n(z_i)) + w_n}{1 + \bar{w}_n a_n(z_i) f_{n-1}(a_n(z_i))} ,$$

so that

$$a_n(z_i) f_{n-1}(a_n(z_i)) = \frac{w_i - w_n}{1 - \bar{w}_n w_i}$$

for all $i = 1, 2, \ldots, n$. Defining

$$\tilde{z}_i = a_n(z_i) \qquad \text{and} \qquad \tilde{w}_i = \frac{w_i - w_n}{1 - \bar{w}_n w_i} \cdot \frac{1 - \bar{z}_n z_i}{z_i - z_n} ,$$

for $i = 1, 2, \ldots, n-1$, we have $|\tilde{z}_i| < 1$, $|\tilde{w}_i| < 1$, and

$$f_{n-1}(\tilde{z}_i) = \tilde{w}_i , \quad i = 1, 2, \ldots, n-1 .$$

This means, by reversing the above procedure, that if we can find an $f_{n-1}(z)$ such that

$$f_{n-1}(\tilde{z}_i) = \tilde{w}_i , \quad i = 1, 2, \ldots, n-1 ,$$

then we obtain

$$f(z) = b_n(a_n(z) \cdot f_{n-1}(a_n(z)))$$

which satisfies $f(z_i) = w_i$, $i = 1, 2, \ldots, n$.

In order to prove the existence of the function $f_{n-1}(z)$, we apply the Interpolation Theorem, together with the given Pick's condition

$$\left[\frac{1 - w_k \bar{w}_l}{1 - z_k \bar{z}_l} \right]_{n \times n} \geqq 0 .$$

In doing so, let $V = V' = \mathscr{C}^{n-1}$, the $(n-1)$-product space of the complex plane $\mathscr{C}$, and define

$$A = \begin{bmatrix} \tilde{z}_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \tilde{z}_{n-1} \end{bmatrix}, \qquad b = \begin{bmatrix} \tilde{w}_1 \\ \vdots \\ \tilde{w}_{n-1} \end{bmatrix}, \qquad \text{and} \qquad c = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} .$$

Then, we have

$$A^j c = [\, \tilde{z}_1^j \ldots \tilde{z}_{n-1}^j \,]^\top , \quad j = 0, 1, \ldots .$$

Consequently, for any $y = [\, y_1 \ldots y_{n-1} \,]^\top \in \mathscr{C}^{n-1}$, we obtain

$$\begin{aligned} \sum_{j=0}^{\infty} |(y, A^j c)|^2 &= \sum_{j=0}^{\infty} \left| \sum_{k=1}^{n-1} \bar{y}_k \tilde{z}_k^j \right|^2 \\ &\leqq \sum_{j=0}^{\infty} \left(\sum_{k=1}^{n-1} |\bar{y}_k|^2 \right) \left(\sum_{k=1}^{n-1} |\tilde{z}_k|^{2j} \right) \\ &= \| y \|_{l^2}^2 \sum_{k=1}^{n-1} \sum_{j=0}^{\infty} |\tilde{z}_k|^{2j} \\ &= \| y \|_{l^2}^2 \sum_{k=1}^{n-1} \frac{1}{1 - |\tilde{z}_k|^2} \\ &< \infty , \end{aligned}$$

so that the condition stated in the Interpolation Theorem is satisfied. Next, we show that the inequality in (ii) of the theorem is also satisfied. Once this is verified,

we have the representation formula in (i) of the theorem, which is

$$\sum_{k=1}^{n-1} \bar{y}_k \tilde{w}_k = (y, b) = \sum_{j=0}^{\infty} h_j (y, A^j c) = \sum_{j=0}^{\infty} h_j \left(\sum_{k=1}^{n-1} \bar{y}_k \tilde{z}_k^j \right)$$

$$= \sum_{k=1}^{n-1} \bar{y}_k \sum_{j=0}^{\infty} h_j \tilde{z}_k^j = \sum_{k=1}^{n-1} \bar{y}_k f_{n-1}(\tilde{z}_k)$$

for all $y = [y_1 \ldots y_{n-1}]^\top \in \mathscr{C}^{n-1}$, where $f_{n-1}(z) := \sum_{j=0}^{\infty} h_j z^j$ is a function in H_1^∞. Obviously, this function $f_{n-1}(z)$ exists in H_1^∞ and satisfies $f_{n-1}(\tilde{z}_k) = \tilde{w}_k$ for $k = 1, 2, \ldots, n-1$, since $y = [y_1 \ldots y_{n-1}]^\top \in \mathscr{C}^{n-1}$ is arbitrary.

To show that the inequality in (ii) of the theorem holds, we carry out all the detailed calculations and obtain the following:

$$\sum_{j=0}^{\infty} |(y, A^j c)|^2 - \sum_{j=0}^{\infty} |(y, A^j b)|^2 = \sum_{j=0}^{\infty} \left[\left| \sum_{k=1}^{n-1} \bar{y}_k \tilde{z}_k^j \right|^2 - \left| \sum_{k=1}^{n-1} \bar{y}_k \tilde{w}_k \tilde{z}_k^j \right|^2 \right]$$

$$= \sum_{j=0}^{\infty} \left[\sum_{k=1}^{n-1} \sum_{l=1}^{n-1} \bar{y}_k y_l \tilde{z}_k^j \bar{\tilde{z}}_l^j (1 - \tilde{w}_k \bar{\tilde{w}}_l) \right]$$

$$= \sum_{k=1}^{n-1} \sum_{l=1}^{n-1} \bar{y}_k y_l (1 - \tilde{w}_k \bar{\tilde{w}}_l) \sum_{j=0}^{\infty} \tilde{z}_k^j \bar{\tilde{z}}_l^j$$

$$= \sum_{k=1}^{n-1} \sum_{l=1}^{n-1} \bar{y}_k \frac{1 - \tilde{w}_k \bar{\tilde{w}}_l}{1 - \tilde{z}_k \bar{\tilde{z}}_l} y_l$$

$$\geqq 0$$

by the Pick condition. Hence, we have

$$\sum_{j=0}^{\infty} |(y, A^j b)|^2 \leqq \sum_{j=0}^{\infty} |(y, A^j c)|^2 ,$$

that is, the inequality in (ii) of the Interpolation Theorem holds. This completes the derivation of the new algorithm.

5. Sequential complexity of the new algorithm

We now calculate the number of (complex) arithmetic operations required by the new Nevanlinna-Pick interpolation algorithm for any given value of z. The result is the following:

Theorem 1. *The new Nevanlinna-Pick interpolation algorithm requires $O(n)$ arithmetic operations to compute $f_n(z)$ for any given value of z.*

Proof. Let $T(n)$ be the number of (complex) arithmetic operations required in computing $f(z) = f_n(z)$ for a given value of z. Since $f_0(z) = 1$ and $f_1(z) = b_1(a_1(z))$, we have $T(0) = 0$ and $T(1) = O(1)$. The number of arithmetic

operations in computing $f(z) = f_n(z)$ can be found by solving the following linear recursion

$$T(n) = O(1) + T(n-1) + O(1) = T(n-1) + O(1)\,. \tag{4}$$

This follows from the observation that the algorithm first computes $a_n(z)$ by performing $O(1)$ arithmetic operations, and then performs a recursive call to compute $f_{n-1}(a_n(z))$ using the currently computed value $a_n(z)$. After executing $T(n-1)$ arithmetic operations, the complex value $f_{n-1}(a_n(z))$ is returned. The algorithm then proceeds to compute $f_n(z) = b_n(a_n(z) \cdot f_{n-1}(a_n(z)))$, which requires an additional $O(1)$ arithmetic steps. Since the solution of (4) gives $T(n) = O(n)$, the total number of (complex) arithmetic operations required by the new Nevanlinna-Pick interpolation algorithm is $O(n)$.

6. Parallel complexity of the new algorithm

Given a particular value of z, we can compute $a_k(z)$ for $1 \leqq k \leqq n$ in parallel. However, the computation of $f_n(z)$ needs the value $f_{n-1}(a_n(z))$, which in turn uses $f_{n-2}(a_{n-1}(a_n(z)))$, and so on. It is not clear if the computation of $f_n(z)$ (as formulated in (3)) can be performed in asymptotically less than $O(n)$ arithmetic operations, given the availability of a pool of processors.

In this section, we give an alternative formulation and show that the new Nevanlinna-Pick interpolation algorithm can be parallelized. The parallel algorithm we propose below is suitable for the shared-memory parallel computer, specifically a CREW PRAM.

Define $f_k(z) = p_k(z)/q_k(z)$ for $1 \leqq k \leqq n$. Since $f_0(z) = 1$, we let $p_0(z) = q_0(z) = 1$. This way, we rewrite (3) as

$$f_k(z) = \frac{p_k(z)}{q_k(z)} = \frac{a_k(z)\dfrac{p_{k-1}(a_k(z))}{q_{k-1}(a_k(z))} + w_k}{1 + \bar{w}_k a_k(z)\dfrac{p_{k-1}(a_k(z))}{q_{k-1}(a_k(z))} + w_k} \tag{5}$$

$$= \frac{a_k(z)p_{k-1}(a_k(z)) + w_k q_{k-1}(a_k(z))}{\bar{w}_k a_k(z)p_{k-1}(a_k(z)) + q_{k-1}(a_k(z))}\,.$$

Let $k = n$, then (5) allows us to represent the numerator and the denominator of $f_n(z)$, namely $p_n(z)$ and $q_n(z)$, by using a matrix recursion of the form

$$\begin{bmatrix} p_n(z) \\ q_n(z) \end{bmatrix} = \begin{bmatrix} a_n(z) & w_n \\ \bar{w}_n a_n(z) & 1 \end{bmatrix} \begin{bmatrix} p_{n-1}(a_n(z)) \\ q_{n-1}(a_n(z)) \end{bmatrix}\,. \tag{6}$$

Similarly, for $k = n-1$, we have

$$\begin{bmatrix} p_{n-1}(z) \\ q_{n-1}(z) \end{bmatrix} = \begin{bmatrix} a_{n-1}(z) & w_{n-1} \\ \bar{w}_{n-1} a_{n-1}(z) & 1 \end{bmatrix} \begin{bmatrix} p_{n-2}(a_{n-1}(z)) \\ q_{n-2}(a_{n-1}(z)) \end{bmatrix}\,. \tag{7}$$

Substituting $z := a_n(z)$ in (7) and then (7) in (6), we can rewrite (6) as

$$\begin{bmatrix} p_n(z) \\ q_n(z) \end{bmatrix} = \begin{bmatrix} a_n(z) & w_n \\ \bar{w}_n a_n(z) & 1 \end{bmatrix} \begin{bmatrix} a_{n-1}(a_n(z)) & w_{n-1} \\ \bar{w}_{n-1} a_{n-1}(a_n(z)) & 1 \end{bmatrix} \begin{bmatrix} p_{n-2}(a_{n-1}(a_n(z))) \\ q_{n-2}(a_{n-1}(a_n(z))) \end{bmatrix}. \tag{8}$$

To proceed, we define the complex numbers c_k for $k = 1, 2, \ldots, n-1$ by the following relation:

$$c_k = a_k(c_{k+1}) \tag{9}$$

with $c_n = a_n(z)$, that is,

$$\begin{aligned} c_n &= a_n(z) \\ c_{n-1} &= a_{n-1}(a_n(z)) \\ c_{n-2} &= a_{n-2}(a_{n-1}(a_n(z))) \\ &\vdots \\ c_1 &= a_1(a_2(a_3(\ldots a_{n-1}(a_n(z))\ldots))). \end{aligned}$$

Moreover, define the 2×2 matrices L_k for $1 \leqq k \leqq n$ by

$$L_k = \begin{bmatrix} c_k & w_k \\ \bar{w}_k c_k & 1 \end{bmatrix}. \tag{10}$$

Then, using the definitions (9) and (10), we can write (6) as

$$\begin{bmatrix} p_n(z) \\ q_n(z) \end{bmatrix} = L_n \begin{bmatrix} p_{n-1}(c_n) \\ q_{n-1}(c_n) \end{bmatrix}.$$

Similarly, Eq. (8) can be written in terms of the matrices L_n and L_{n-1} as

$$\begin{bmatrix} p_n(z) \\ q_n(z) \end{bmatrix} = L_n L_{n-1} \begin{bmatrix} p_{n-2}(c_{n-1}) \\ q_{n-2}(c_{n-1}) \end{bmatrix}.$$

Continuing this iteration, we finally obtain $p_n(z)$ and $q_n(z)$ in terms of L_k for $1 \leqq k \leqq n$ as follows:

$$\begin{bmatrix} p_n(z) \\ q_n(z) \end{bmatrix} = L_n L_{n-1} \ldots L_2 L_1 \begin{bmatrix} p_0(c_1) \\ q_0(c_1) \end{bmatrix}.$$

Since $p_0(c_1) = q_0(c_1) = 1$ by definition, we finally arrive at

$$\begin{bmatrix} p_n(z) \\ q_n(z) \end{bmatrix} = L_n L_{n-1} \ldots L_2 L_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{11}$$

The new definition of $f_n(z)$ given by (11) allows parallel computation since the product of n 2×2 matrices can be computed in $O(\log n)$ arithmetic operations on

a CREW PRAM with n processors via the well-known associative fan-in algorithm (the binary tree multiplication algorithm) [1, 9]. This technique, however, requires the formation of the matrices L_k for $1 \leqq k \leqq n$ in parallel, which requires the parallel computation of the quantities c_k defined by the recursive relation (9). In the following, we show that the computation of c_k for $1 \leqq k \leqq n$ can also be performed in parallel.

We have $c_n = a_n(z)$ and for $k = n-1, n-2, \ldots, 2, 1$,

$$c_k = a_k(c_{k+1}) = \frac{c_{k+1} - z_k}{1 - \bar{z}_k c_{k+1}}.$$

By defining $c_k = r_k/s_k$ for $1 \leqq k \leqq n-1$, $r_n = c_n = a_n(z)$, and $s_n = 1$, the above can be written as

$$c_k = \frac{r_k}{s_k} = \frac{\dfrac{r_{k+1}}{s_{k+1}} - z_k}{1 - \bar{z}_k \dfrac{r_{k+1}}{s_{k+1}}} = \frac{r_{k+1} - s_{k+1} z_k}{-\bar{z}_k r_{k+1} + s_{k+1}}$$

for $1 \leqq k \leqq n-1$. This formulation yields a matrix recursion for r_k and s_k (similar to (6)) as follows:

$$\begin{bmatrix} r_k \\ s_k \end{bmatrix} = M_k \begin{bmatrix} r_{k+1} \\ s_{k+1} \end{bmatrix}, \tag{12}$$

where $1 \leqq k \leqq n-1$ and M_k is a 2×2 matrix defined by

$$M_k = \begin{bmatrix} 1 & -z_k \\ -\bar{z}_k & 1 \end{bmatrix}.$$

It follows immediately that

$$\begin{bmatrix} r_k \\ s_k \end{bmatrix} = M_k M_{k+1} \cdots M_{n-1} \begin{bmatrix} r_n \\ s_n \end{bmatrix} = M_k M_{k+1} \cdots M_{n-1} \begin{bmatrix} a_n(z) \\ 1 \end{bmatrix}. \tag{13}$$

Consequently, all the c_k for $1 \leqq k \leqq n-1$ are obtained by computing the prefix products of the matrices M_k for $1 \leqq k \leqq n-1$. Namely, $c_n = a_n(z)$ and $c_k = r_k/s_k$ for $k = n-1, n-2, \ldots, 2, 1$, where

$$\begin{bmatrix} r_{n-1} \\ s_{n-1} \end{bmatrix} = M_{n-1} \begin{bmatrix} a_n(z) \\ 1 \end{bmatrix},$$

$$\begin{bmatrix} r_{n-2} \\ s_{n-2} \end{bmatrix} = M_{n-1} M_{n-2} \begin{bmatrix} a_n(z) \\ 1 \end{bmatrix},$$

$$\vdots$$

$$\begin{bmatrix} r_1 \\ s_1 \end{bmatrix} = M_{n-1} M_{n-2} \cdots M_2 M_1 \begin{bmatrix} a_n(z) \\ 1 \end{bmatrix}.$$

We summarize this parallel Nevanlinna-Pick interpolation algorithm in the following:

The new parallel Nevanlinna-Pick algorithm

Step 1. Given z and z_k for $1 \leqq k \leqq n$, compute $a_k(z)$ for $1 \leqq k \leqq n$ by

$$a_k(z) = \frac{z - z_k}{1 - \bar{z}_k z}.$$

Step 2. For $1 \leqq k \leqq n-1$, set

$$M_k = \begin{bmatrix} 1 & -z_k \\ -\bar{z}_k & 1 \end{bmatrix},$$

and then compute

$$N_{n-1} = M_{n-1}$$
$$N_{n-2} = M_{n-1} M_{n-2}$$
$$\vdots$$
$$N_1 = M_{n-1} M_{n-2} \cdots M_1.$$

Step 3. Set $c_n = a_n(z)$. Compute r_k and s_k for $1 \leqq k \leqq n-1$ by

$$\begin{bmatrix} r_k \\ s_k \end{bmatrix} = N_k \begin{bmatrix} a_n(z) \\ 1 \end{bmatrix},$$

and then compute $c_k = r_k / s_k$ for $1 \leqq k \leqq n-1$.

Step 4. For $1 \leqq k \leqq n$, *set*

$$L_k = \begin{bmatrix} c_k & w_k \\ \bar{w}_k c_k & 1 \end{bmatrix},$$

and then compute

$$L = L_n L_{n-1} \ldots L_2 L_1.$$

Step 5. Compute $p_n(z)$ and $q_n(z)$ using

$$\begin{bmatrix} p_n(z) \\ q_n(z) \end{bmatrix} = L \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

and then set $f(z) = f_n(z) = p_n(z)/q_n(z)$.

Next, we calculate the number of parallel arithmetic operations required by the above algorithm.

Theorem 2. *For any given value of z, the new parallel Nevanlinna-Pick interpolation algorithm requires $O(\log n)$ arithmetic operations on a CREW PRAM with n processors.*

Proof. The complex numbers $a_k(z)$ for $1 \leqq k \leqq n$ in Step 1 can be computed in parallel by allocating a single processor for each $a_k(z)$. Thus, n processors compute $a_k(z)$ for all $k = 1, 2, \ldots, n$ using $O(1)$ parallel arithmetic operations.

Once the matrices M_k for $1 \leqq k \leqq n-1$ are formed, we need to compute the prefix products of $n-1$ 2×2 matrices. There exist several parallel algorithms for the prefix computation problem. It is well-known that the prefix product of n quantities can be performed in $O(\log n)$ operations with n processors [4, 6–8, 12]. An algorithm suitable for implementation on a CREW PRAM is described in [7]. Since the quantities involved in the above version of the prefix computation are 2×2 matrices, we conclude that the computation of N_k for $1 \leqq k \leqq n-1$ requires $O(\log n)$ arithmetic operations with $n-1$ processors.

The computation of r_k and s_k for $1 \leqq k \leqq n$ required in Step 3 can be performed in parallel in $O(1)$ arithmetic operations with n processors. This can be achieved by assigning a single processor for each matrix-vector product. Each of the n processors then computes c_k for $1 \leqq k \leqq n$ by performing a single division operation.

Similarly, Step 4 requires a single complex multiplication for each of the n matrices L_k for $1 \leqq k \leqq n$ with n processors. The computation of L can be achieved by using at most $\lfloor n/2 \rfloor$ processors in $O(\log n)$ arithmetic operations with the help of the associative fan-in algorithm [1, 9]. This algorithm makes use of a binary tree of depth $\lceil \log_2 n \rceil$ which contains the matrices L_k for $1 \leqq k \leqq n$ as its leaves, and whose root has the final value $L = L_1 L_2 \cdots L_n$.

Finally, the computation of $p_n(z)$ and $q_n(z)$ and hence $f_n(z)$ requires $O(1)$ arithmetic operations using a single processor.

We thus come to the conclusion that the parallel version of the new Nevanlinna-Pick interpolation algorithm requires at most n processors, and the total number of arithmetic operations is bounded by $O(\log n)$. □

In case we have only $p < n$ processors, we can compute $a_k(z)$ for $1 \leqq k \leqq n$ *in* $O(m)$ arithmetic steps by assigning a single processor to compute

$$a_{(i-1)m+1}(z), \quad a_{(i-1)m+2}(z), \ldots, a_{(i-1)m+m}(z)$$

where $m = \lceil n/p \rceil$. Processors $i = 1, 2, 3, \ldots, p-1$ compute $a_k(z)$ for $1 \leqq k \leqq (p-1)m$ while the last processor computes $a_k(z)$ for $(p-1)m + 1 \leqq k \leqq n$. The total number of parallel arithmetic operations is bounded by $O(m) = O(n/p)$. Similarly, Step 3 takes $O(n/p)$ arithmetic operations.

The product of n elements requires $O(n/p + \log p)$ arithmetic steps using $p < n$ processors. The computation is performed by grouping n elements into p groups, each group containing $m = \lceil n/p \rceil$ elements. We assign a single processor to each group and simultaneously compute all the group products using $O(m)$ arithmetic operations. The final product is computed by multiplying all p group products using p processors, which takes an additional $O(\log p)$ arithmetic operations. It is also known that the prefix product of n elements can be computed by performing $O(n/p + \log p)$ parallel arithmetic operations with $p < n$ processors [7]. Thus, Steps 2 and 4 take $O(n/p + \log p)$ parallel arithmetic steps with p processors. This gives us the following result:

Theorem 3. *For any given value of z, the new parallel Nevanlinna-Pick interpolation algorithm requires $O(n/p + \log p)$ arithmetic operations on a CREW PRAM with $p < n$ processors.*

7. Conclusion

We have proposed a new algorithm for the Nevanlinna-Pick interpolation which differs from the classical sequential and parallel schemes studied by the present authors [5] in that the new algorithm avoids the computation of the Fenyves array and evaluates the interpolatory rational function directly at any given point by using a very simple and straightforward recursive formula. The new Nevanlinna-Pick interpolation algorithm requires only $O(n)$ arithmetic operations to evaluate the interpolatory rational function at any given point, and yields a parallel scheme requiring only $O(\log n)$ arithmetic operations on a CREW PRAM with n processors.

References

1. Bertsekas, D.P., Tsitsiklis, J.N. (1989): Parallel and distributed computation, Numerical Methods. Prentice-Hall, Englewood Cliffs, N.J.
2. Chui, C.K., Chen. G. (1992): Signal processing and systems theory: Selected topics. Springer, Berlin Heidelberg New York
3. Fortune, S., Wyllie, J. (1978): Parallelism in random access machines. In: Proceedings of the ACM Symposium on the Theory of Computing, pp. 114–118, San Diego, Cal.
4. Greenberg, A.G., Ladner, R.E., Paterson, M., Galil, Z. (1982): Efficient parallel algorithms for linear recurrence computation. Inf. Proc. Lett. **15**, 31–35
5. Koç, Ç.K., Chen, G. (1991): Parallel algorithms for Nevanlinna-Pick interpolation: The scalar case. Int. J. Comput. Math. **40**, 99–115
6. Kogge, P.M., Stone, H.S. (1973): A parallel algorithm for the efficient solution of a general class of recurrence equations. IEEE Trans. Comput. **22**, 786–792
7. Kruskal, C.P., Rudolph, L., Snir, M. (1985): The power of parallel prefix. IEEE Trans. Comput. **34**, 965–968
8. Ladner, R., Fischer, M. (1980): Parallel prefix computation. J. ACM **27**, 831–838
9. Lakshmivarahan, S., Dhall, S.K. (1990): Analysis and design of parallel algorithms. McGraw-Hill, New York, N.Y.
10. Rosenblum, M., Rovnyak, J. (1985): Hardy classes and operator theory. Oxford University Press, Oxford
11. Snir, M. (1985): On parallel searching. SIAM J. Comput. **14**, 688–708
12. Snir, M. (1986): Depth-size trade-offs for parallel prefix computation. J. Algorithms **7**, 185–201