NEW ALGORITHMS FOR POLYNOMIAL AND TRIGONOMETRIC INTERPOLATION ON PARALLEL COMPUTERS

by

Ilan Bar-On and Avram Sidi

Technical Report #660

December 1990
New Algorithms for Polynomial and Trigonometric Interpolation on Parallel Computers

Ilan Bar-On and Avram Sidi
Technion - Israel Institute of Technology,
Department of Computer Science,
Technion City, Haifa 32 000, Israel.

December 19, 1990
Abstract

An interpolation polynomial of order $N$ is constructed from $p$ independent subpolynomials of order $n \sim N/p$. Each such subpolynomial is found independently and in parallel. Moreover, evaluation of the polynomial at any given point is done independently and in parallel, except for a final step of summation of $p$ elements. Hence, the algorithm has almost no communication overhead and can be implemented easily on any parallel computer. We give examples of finite-difference interpolation, trigonometric interpolation, and Chebyshev interpolation, and conclude with the general Hermite interpolation problem.
1 Introduction

In this paper we study the problem of polynomial and trigonometric interpolation on large parallel MIMD computers. There are well known sequential methods for both problems. These include Lagrange, Newton, and finite-difference formulas for polynomial interpolation, and variations of the FFT algorithm for Chebyshev and trigonometric interpolation[5, 8]. However, these methods are not easily adaptable to parallel systems, and especially to loosely connected systems such as rings, stars[6] because of the overhead due to interprocessor communication.

In this work we present new algorithms for polynomial and trigonometric interpolation that require almost no communication between the processors. Given an interpolation problem of order \( N = np \), we subdivide it into \( p \) smaller interpolation problems of order \( n \), of the same type as the original problem. More precisely, given a finite difference interpolation problem, we subdivide it into smaller finite difference interpolation problems, see Section 3. Given a trigonometric interpolation problem that can be solved using the FFT algorithm we subdivide it into smaller trigonometric interpolation problems that can be solved using the FFT algorithm, see Section 4. In Section 5 we show how to subdivide the Chebyshev interpolation problem into smaller similar subproblems, and in Section 6, we consider the general Hermite interpolation problem.

In Section 2 we present the interpolation polynomial which may be thought of as a compromise between Newton and Lagrange formulas. Given a set of
interpolation points, we subdivide them among the processors so that each has to find a subpolynomial on a smaller set of points. These subpolynomials can be found independently and in parallel by all the processors with no interprocessor communication overhead at all. Each processor may employ an arbitrary sequential method that is suitable numerically and computationally. There is no need for all processors to use the same interpolation method or the same number of points, although it is recommended that they all solve their subproblems in approximately the same time. To evaluate the polynomial we add the values of the corresponding subpolynomials. We observe that in the case of a single processor we are simply back to the sequential case, whereas in the case of \( N \) processors, \( N \) equals the degree of the interpolation problem, we get the Lagrange method. Hence, our algorithm adapts itself to the number of processors available. For example, consider the star architecture depicted in the figure. Here, each processor may compute in parallel the function values at its interpolation points and then construct its subpolynomial independently to the other processors. In evaluating the interpolation polynomial, each processor first evaluates its subpolynomial at the given point and then sends its result to the master processor which computes the final result. Hence, in most practical cases where \( p \ll N \) this last step will contribute little to the overall complexity of the algorithm. We conclude that the algorithm can be efficiently implemented on almost any parallel system.

Recently Eğecioğlu and Gallopoulos[3] suggested a parallel implementation of the Newton formula using prefix circuits. Reif[7] gives fast arithmetic circuits for computing the polynomial, and Dowling[2] presents a fast parallel Horner algorithm. However, these methods are not applicable to large MIMD computers because of the large communication overhead.

## 2 The interpolation polynomial

Let \( X = \{x_0, x_1, \ldots, x_N \} \) be given distinct points in the interval \([a, b]\), and let \( f(x) \) be a function defined on \([a, b]\), whose values \( f_j = f(x_j), j = 0, 1, \ldots, N \), are given. We are interested in constructing a representation of the polynomial \( P(x) \) of degree at most \( N \) that interpolates \( f(x) \) on \( X \) and is most suitable for parallel computation.
2 THE INTERPOLATION POLYNOMIAL

Let \( \{X_1, X_2, \ldots, X_p\} \) be a partition of \( X \), i.e.,

\[
X = \bigcup_i X_i \quad \text{and} \quad X_i \cap X_j = \emptyset \quad \text{for} \quad i \neq j.
\]

The following theorem indicates how \( P(x) \) can be constructed independently and in parallel by \( p \) processors, each solving a smaller interpolation problem on one of the subsets \( X_i \).

**Theorem 2.1** For \( i = 1, \ldots, p \), define

\[
\omega_{ij} = \prod_{k \in X_i} (x_j - x_k), \quad x_j \in X_i,
\]

and let \( Q_i(x) \) be the polynomial of degree at most \( |X_i| - 1 \) that satisfies the following interpolation conditions:

\[
Q_i(x_j) = \omega_{ij} f_j, \quad x_j \in X_i.
\]

Then \( P(x) \), the interpolation polynomial on \( X \), is given by

\[
P(x) = \sum_{i=1}^{p} Q_i(x) \prod_{z \in X_i} (x - z).
\]

**Proof:** First, it is clear that the right hand side of (4) is a sum of polynomials, each of degree at most \( N \). Next, with the help of (2) and (3), it can be shown that \( P(x) \) satisfies the interpolation conditions

\[
P(x_j) = f(x_j) \quad \text{for all} \quad j.
\]

The result now follows from the uniqueness of \( P(x) \). \( \square \)

It is known\[4, 9, 10\] that the barycentric representation for Lagrange interpolation enjoys a large degree of numerical stability. We, therefore, look for a generalization of the barycentric representation that is appropriate for the formula given in (4). Precisely this is achieved in Theorem (2.2) below.

**Theorem 2.2** Let \( Q_i(x) \) be as in Theorem (2.1), and let \( R_i(x) \), \( i = 1, \ldots, p \), be the polynomial of degree at most \( |X_i| - 1 \) that satisfies the interpolation conditions

\[
R_i(x_j) = w_{ij} f_j, \quad x_j \in X_i.
\]

Then \( P(x) \) can be expressed in the form

\[
P(x) = \frac{\sum_{i=1}^{p} Q_i(x) / \prod_{z \in X_i} (x - z)}{\sum_{i=1}^{p} R_i(x) / \prod_{z \in X_i} (x - z)} \equiv \frac{\sum_{i=1}^{p} \phi_i(x)}{\sum_{i=1}^{p} \psi_i(x)}.
\]

Technion - Computer Science Department - Technical Report CS0660 - 1990
Proof: Comparing (3) and (6), and employing (4), we see that

\[ 1 = \sum_{i=1}^{p} R_i(x) \prod_{x_k \in X_i} (x - x_k) \quad \text{for all } x, \]

which we rewrite in the form

\[ 1 = \prod_{x_k \in X} (x - x_k) \sum_{i=1}^{p} R_i(x) / \prod_{x_k \in X_i} (x - x_k). \]

Similarly,

\[ P(x) = \prod_{x_k \in X} (x - x_k) \sum_{i=1}^{p} Q_i(x) / \prod_{x_k \in X_i} (x - x_k). \]

The result now follows by dividing (10) by (9). □

Given \( p \) processors, we assign processor \( i = 1, \ldots, p \), to computing the corresponding terms \( \psi_i(x) \) and \( \psi_i(x) \). The computation of the \( w_{i,j} \) take \( O(n_i(N - n_i)) \) additions and multiplications in the worst case, where \( n_i = |X_i| \). However, as will be seen in the following sections, in many cases of interest these values can be computed analytically in much fewer operations. Assuming that the \( w_{i,j} \) are known, and that \( n_i \sim n \sim N/p, i = 1, \ldots, p \), each processor is faced with an interpolation problem of order \( n \) that can be solved in parallel to the other problems with no need of interprocessor communication. Once the interpolation polynomial is known, its value at points not in the set are given by summing and dividing the corresponding subvalues in (7). Theoretically, we may compute this final result in approximately \( \log p \) parallel steps of additions and communications[1]. However, this requires that the corresponding interprocessor links are available, which is not the case for loosely connected systems such as rings, stars etc. Even in tightly connected systems, when \( p \) is not too large, this last step will be done more efficiently using a single sequential processor. Parallel systems usually have a main processor to which all other processors are directly connected. Each processor can sends its computed value to the main processor in a single communication step. The main processor then computes the final result. We conclude that our interpolation polynomial is most suitable for parallel systems.

In Sections 3, 4, 5 we consider the problems of finite difference interpolation, trigonometric interpolation, and Chebyshev interpolation. For ease of
representation we will use a slightly different notation as follows: We assume that the function $f(x)$ is given at $N = np$ distinct points and that each of the $p$ subsets $X_i$, which are now numbered by $i = 0, \ldots, p - 1$, contains exactly $n$ points. We denote the points in the subset $X_i$ by $z_{i,j}$, $j = 0, \ldots, n - 1$.

3 Finite Difference Interpolation

Let $X$ be a set of equally spaced points in the interval $[a, b]$, 

$$z_i = a + ih, \quad i = 0, 1, \ldots, N - 1, \quad h = \frac{b - a}{N - 1},$$

where we assume for simplicity that $N = np$ and $p$ is the number of processors available. We consider here two partitions.

In the first partition we assign the $i$th group of $n$ consecutive points to the $i$th processor, i.e.,

$$X_i = \{z_{i,j} = z_{in+j}, \quad j = 0, \ldots, n - 1\}, \quad i = 0, \ldots, p - 1.$$

Hence,

$$w_{i,j}^{-1} = \prod_{k=0}^{i-1}(z_{in+j} - z_k) \prod_{k=(i+1)n}^{N-i}(z_{in+j} - z_k)$$

$$= C_i(-1)^{in}\left(\frac{n-1}{j}\right)/(N-1),$$

where $C_i = (-h)^{N-n(N-1)!/(n-1)!}$ is independent of $i$ and $j$. We note that if the same constant $C$ multiplies all the $w_{i,j}$, it follows from (3),(6) that the subpolynomials $Q_i(x)$ and $R_i(x)$ are also multiplied by the same constant $C$ but the interpolation polynomial $P_n(x)$ remains invariant by (7). In view of this, each processor has to compute the corresponding polynomials $Q_i(x)$ and $R_i(x)$ that satisfy the interpolation conditions

$$Q_i(z_{i,j}) = (-1)^{in}f_{in+j}\left(\frac{N-1}{in+j}\right)/(n-1),$$

$$R_i(z_{i,j}) = (-1)^{in}\left(\frac{N-1}{in+j}\right)/(n-1),$$
for \( j = 0, \ldots, n - 1 \), and this computation can be carried out using any finite difference formula.

In the second partition the subsets \( X_i \) are formed according to

\[
X_i = \{ z_{i,j} = z_{i+jp}, \quad j = 0, \ldots, n - 1 \}, \quad i = 0, \ldots, p - 1.
\]

Consequently,

\[
w_{i,j}^{-1} = \frac{\prod_{k=0}^{i+jp-1}(z_{i+jp} - z_k) \prod_{k=i+jp+1}^{N-1}(z_{i+jp} - z_k)}{\prod_{k=0}^{i+jp}(z_k - z_{i+jp}) \prod_{k=i+jp+1}^{N-1}(z_{i+jp} - z_{i+jp})}
\]

\[
= C_2(-1)^{i+j(p-1)} \binom{n-1}{j} / \binom{N-1}{i+jp},
\]

where \( C_2 = C_1/p^{p-1} \) is independent of \( i \) and \( j \). Each processor has to compute the corresponding polynomials \( Q_i(z) \) and \( R_i(z) \) that satisfy the interpolation conditions

\[
Q_i(z_{i,j}) = (-1)^{i+j(p-1)} f_{i+jp} \binom{N-1}{i+jp} / \binom{n-1}{j},
\]

\[
R_i(z_{i,j}) = (-1)^{i+j(p-1)} \binom{N-1}{i+jp} / \binom{n-1}{j},
\]

for \( j = 0, \ldots, n - 1 \). As before, this computation can be carried out using any finite difference formula.

We have the following operation count for constructing and evaluating the polynomial, when \( p \ll N \):

<table>
<thead>
<tr>
<th>Operation</th>
<th>Sequential</th>
<th>Parallel</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construction</td>
<td>((N-1)N/2)</td>
<td>((n-1)n)</td>
<td>(\sim p^2/2)</td>
</tr>
<tr>
<td>Evaluation</td>
<td>(N)</td>
<td>(2(n+p))</td>
<td>(\sim p/2)</td>
</tr>
</tbody>
</table>

We obtain a speed-up of order \( p^2/2 \) in the construction stage, and a speed-up of order \( p/2 \) in the evaluation stage as compared to the ordinary sequential finite difference methods.
4 Trigonometric Interpolation

Let \( \theta_j, j = 0, 1, \ldots, N - 1 \), be equally spaced points in \([0, 2\pi]\) given by

\[
\theta_j = \frac{2\pi j}{N}, \quad j = 0, 1, \ldots, N - 1, \tag{21}
\]

and let \( f(\theta) \) be a function defined on \([0, 2\pi]\) whose values \( f_j \equiv f(\theta_j), j = 0, 1, \ldots, N - 1 \), are given. Furthermore, let \( N = 2M \). Then there exists a unique balanced trigonometric polynomial \( T(\theta) \) of degree \( M \),

\[
T(\theta) = \frac{1}{2}a_0 + \sum_{k=1}^{M-1} (a_k \cos k\theta + b_k \sin k\theta) + \frac{1}{2}a_M \cos M\theta \tag{22}
\]

interpolating \( f(\theta) \) at the points \( \theta_j, j = 0, 1, \ldots, N - 1 \), see[4]. A complex interpretation of \( T(\theta) \) in terms of the variable \( z = e^{i\theta} \) yields the balanced complex trigonometric polynomial \( P(z) \equiv T(\theta) \) of degree \( M \),

\[
P(z) = \frac{1}{2}c_M z^{M} + \sum_{k=1}^{M-1} c_k z^{k} + \frac{1}{2}c_M z^{-M}, \quad c_{-M} = c_M, \tag{23}
\]

whose coefficients \( c_k \) are related to the \( a_k \) and \( b_k \) in (22) through

\[
a_k = c_k + c_{-k}, \quad b_k = i(c_k - c_{-k}), \quad k = 0, 1, \ldots, M. \tag{24}
\]

Of course, \( P(z) \) satisfies the interpolation conditions

\[
P(z_j) = T(\theta_j) = f_j, \quad j = 0, 1, \ldots, N - 1, \tag{25}
\]

where the \( z_j \) are given by

\[
z_j = z_1^j, \quad j = 0, 1, \ldots, N - 1, \quad \text{with} \quad z_1 = e^{i2\pi/N}. \tag{26}
\]

The coefficients \( c_l \) of \( P(z) \) can be computed from

\[
c_l = \frac{1}{N} \sum_{k=0}^{N-1} f_k z_1^{l}, \quad l = -M, -M + 1, \ldots, M, \tag{27}
\]

and this, with the help of the FFT, can be done in \( O(N \log N) \) operations. However, parallelization of the FFT algorithm is not a simple task and nor is...
that of evaluating the resulting polynomial at arbitrary points. In this section we introduce a new representation for $P(z)$ that can be computed on a parallel computer with almost no interprocessor communication. Specifically, we divide the original trigonometric interpolation problem into $p$ trigonometric interpolation problems of smaller size, each of which is of the same type as the original problem. Furthermore, the FFT can be employed in each of these problems.

Let $N = np$ with $n = 2^m$, and consider the partition $\{Z_0, Z_1, \ldots, Z_{p-1}\}$ of the set of points $Z = \{z_0, z_1, \ldots, z_{N-1}\}$, where

$$Z_\ell = \{z_{\ell+r} = z_{\ell+r}, \quad r = 0, 1, \ldots, n-1\}, \quad \ell = 0, 1, \ldots, p-1.$$ 

**Theorem 4.1** For $\ell = 0, 1, \ldots, p-1$, define

$$w^{-1}_{r,\ell} = z_{r,\ell}^{M+m} \prod_{k \neq \ell} (z_{r,\ell} - z_{k,\ell}), \quad r = 0, 1, \ldots, n-1,$$

and let $Q_{\ell}(s)$ be the balanced complex trigonometric polynomial of degree $m$ that satisfies the interpolation conditions

$$Q_{\ell}(z) = f_{r,\ell} w_{r,\ell}, \quad r = 0, 1, \ldots, n-1,$$

on the subset of points $Z_\ell$. Then $P(z)$, the balanced trigonometric polynomial that satisfies the interpolation conditions in (25), can be expressed in the form

$$P(z) = z^{-M+m} \sum_{\ell=0}^{p-1} Q_{\ell}(z) \prod_{k \neq \ell} (z - z_{k,\ell})$$

**Proof:** First, it is clear from (29) and (30) that $P(z)$ in (31) satisfies the interpolation conditions in (25). Next, by (23), $Q_{\ell}(s)$ is of the form:

$$Q_{\ell}(s) = \frac{1}{2} d_{-m} s^{-m} + \sum_{k=-m+1}^{m-1} d_s s^k + \frac{1}{2} d_{-m} s^m, \quad d_{-m} = d_m.$$

Also,

$$z^{-M+m} \prod_{k \neq \ell} (z - z_{k,\ell}) = \sum_{k=-M+m}^{M-m} g_k z^k$$
4 TRIGONOMETRIC INTERPOLATION

with

\[ g_{-M+m} = \prod_{s \neq l} (-z_{s,l}) = \prod_{s=0}^{n-1} \frac{(-z_j)}{(-z_{s,l})} = z_i^{-n} \quad \text{and} \quad g_{M-m} = 1. \]

The first result in (34) follows from the fact that the \( z_j \) are \( N \)th roots of unity, while the \( z_s \) are the \( n \)th roots of unity. Substituting (32) in (31), and using (33) and (34), we see that each of the terms \( z^{-M+m}Q(z/z_l) \prod_{s \neq l}(z - z_{s,l}) \) in (31) is of the form

\[ h_{-M} = \frac{1}{2} d_{-m} z_i^m g_{M+m} = \frac{1}{2} d_{-m} z_i^{-m} = \frac{1}{2} d_{-m} z_i^{-m} g_{M-m} = h_M, \]

and this ensures that \( P(z) \) is balanced. The rest follows from the uniqueness of \( P(z) \).

As before, we look for a generalized barycentric formula. This formula is developed in Theorem 4.2 below.

**Theorem 4.2** For \( i = 0,1,\ldots,p-1 \), let \( \hat{Q}_i(s) \), be the balanced complex trigonometric polynomial of degree \( m \) that satisfies the interpolation conditions

\[ \hat{Q}_i(z_i^p) = (-1)^i (p^{-1}) f_{i+p}, \quad r = 0,1,\ldots,n-1, \]

on the subset of points \( Z_0 \). Then the balanced trigonometric interpolation polynomial \( P(z) \) of Theorem (4.1) can be expressed in the form

\[ P(z) = \frac{\sum_{i=0}^{n-1} (-1)^i z_i^{-m} \hat{Q}_i(z/z_1)/((z/z_1)^n - 1)}{\sum_{i=0}^{n-1} (-1)^i z_i^{-m} \hat{R}(z/z_1)/((z/z_1)^n - 1)}, \]

where, depending on whether \( p \) is even or odd, \( \hat{R}(s) \) assumes the simple forms

\[ \hat{R}(s) = \begin{cases} 
1 & \text{if } p \text{ is odd}, \\
\frac{1}{2} (s^n + s^{-m}) & \text{if } p \text{ is even}.
\end{cases} \]

**Proof:** For \( i = 0,1,\ldots,p-1 \), we let \( R_i(s) \) be the balanced complex trigonometric polynomial of degree \( m \) that satisfies the interpolation conditions

\[ R_i(z_i^p) = w_i, \quad r = 0,1,\ldots,n-1, \]

where

\[ w_i = \frac{\sum_{j=0}^{n-1} (-1)^j z_j^{-m} \hat{Q}_j(z/z_1)/((z/z_1)^n - 1)}{\sum_{j=0}^{n-1} (-1)^j z_j^{-m} \hat{R}(z/z_1)/((z/z_1)^n - 1)}, \]
TRIGONOMETRIC INTERPOLATION

Comparing (39) with (30), it is obvious from Theorem 4.1 that

\begin{equation}
1 \equiv z^{-M+m} \sum_{k=0}^{p-1} R_i(z/z_l) \prod_{k \neq i} (z - z_{k,l}).
\end{equation}

Dividing now (31) by (40), we obtain the barycentric formula for \( P(z) \)

\begin{equation}
P(z) = \frac{\sum_{k=0}^{p-1} Q_i(z/z_l)/\prod_{i=0}^{p-1}(z - z_{i,r})}{\sum_{k=0}^{p-1} R_i(z/z_l)/\prod_{i=0}^{p-1}(z - z_{i,r})}.
\end{equation}

Next, we observe that

\begin{equation}
\prod_{k \neq l}(z_{l,r} - z_{k,l}) = \prod_{i \neq r}(z_{i,r} - z_{k,i})/\prod_{i \neq r}(z_{i,r} - z_{l,i})
= (Nz_{i,r}^{N-1})/(u z_{i,r}^{m-1}) = p z_{i,r}^{-m},
\end{equation}

so that (29) becomes

\begin{equation}
\omega_{i,r}^{-1} = p z_{i,r}^{-M+m} = p(-1)^{l+r(m-1)} z_{i,r}^{-m}.
\end{equation}

Furthermore,

\begin{equation}
\prod_{r=0}^{n-1}(z - z_{i,r}) = \prod_{r=0}^{n-1}(z/z_l - z_i^*) = z_i^n((z/z_l)^m - 1).
\end{equation}

Comparing (36) with (30), and invoking (43), we see that

\begin{equation}
Q_i(s) = p(-1)^l z_i^n \bar{Q}_i(s), \quad l = 0, 1, \ldots, p - 1.
\end{equation}

Similarly, if we define \( \hat{R}(s) \) to be the balanced complex trigonometric polynomial that satisfies the interpolation conditions

\begin{equation}
\hat{R}(z_i^*) = (-1)^{(p-1)}, \quad r = 0, 1, \ldots, n - 1,
\end{equation}

then

\begin{equation}
\hat{R}_l(s) = p(-1)^l z_i^n \hat{R}(s), \quad l = 0, 1, \ldots, p - 1.
\end{equation}

Combining (44),(45), and (47) in (41), we obtain (37). Finally, (38) can be seen to hold by inspection. \( \square \)
Now that we have obtained the barycentric form of $P(z)$, we can obtain that of $T(\theta)$, the real form of $P(z)$, very easily as follows:

\[
T(\theta) = \sum_{m=0}^{\infty} (-1)^m \frac{U_m(\theta - \theta_1)}{\sin m(\theta - \theta_1)} \sum_{j=0}^{\infty} (-1)^j V_j(\theta - \theta_1) \frac{\sin m(\theta - \theta_1)}{\sin m(\theta - \theta_1)},
\]

where $U_m(\theta) = Q_m(\theta)$, with $\theta = e^{i \phi}$, is the balanced trigonometric polynomial of degree $m$ that satisfies the interpolation conditions

\[
U_m(\theta_j) = (-1)^{j-1} f_{j-1}, \quad j = 0, 1, \ldots, n - 1,
\]

and $\hat{V}(\phi)$ is given by

\[
\hat{V}(\phi) = \begin{cases} 
1 & \text{if } p \text{ is odd.} \\
\cos m\phi & \text{if } p \text{ is even.}
\end{cases}
\]

We have the following operation count for constructing and evaluating the polynomial, when $p \ll N$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Sequential</th>
<th>Parallel</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construction</td>
<td>$N \log N$</td>
<td>$n \log n$</td>
<td>$\sim p$</td>
</tr>
<tr>
<td>Evaluation</td>
<td>$N$</td>
<td>$n + 2p$</td>
<td>$\sim p$</td>
</tr>
</tbody>
</table>

We obtain a speed-up of order $p$ both in the construction and evaluation of the polynomial as compared to the sequential FFT algorithm.

5 Chebyshev interpolation

Let $x_j, j = 0, 1, \ldots, N - 1$, be $N$ Chebyshev points in $[-1, 1]$ given by

\[
x_j = \cos \theta_j, \quad \theta_j = \frac{2j + 1}{2N} \pi, \quad j = 0, 1, \ldots, N - 1,
\]

and let $f(x)$ be a function defined on $[-1, 1]$ whose values $f_j = f(x_j), j = 0, 1, \ldots, N - 1$, are given. Let $N = np$, and consider the partition $X = \{X_0, X_1, \ldots, X_{p-1}\}$, where

\[
X_l = \{x_{l+r}, \quad r = 0, 1, \ldots, n - 1\}, \quad l = 0, 1, \ldots, p - 1.
\]

We define a new partition, $Y = \{Y_0, Y_1, \ldots, Y_{q-1}\}, q = \lfloor (p+1)/2 \rfloor$ as follows:

\[
Y_l = X_l \cup X_{l'}, \quad l' = p - 1 - l, \quad l = 0, 1, \ldots, q - 1.
\]
Lemma 5.1 For \( \ell = 0, 1, \ldots, q - 1 \), define
\[
\omega_{\ell,r} = \prod_{k \neq \ell} (z_{\ell,r} - x_{k,t}), \quad r = 0, 1, \ldots, n - 1,
\]
and let \( Q_\ell(x) \) be the polynomial of degree \(|Y_i| - 1\) that satisfies the interpolation conditions
\[
Q_\ell(x_{k,r}) = f_{k+\ell} w_{k,r}, \quad k = 1, l', \quad r = 0, 1, \ldots, n - 1,
\]
on the subset of points \( Y_i \). Then \( P(x) \), the interpolation polynomial on \( X \), can be expressed in the form
\[
P(x) = \sum_{\ell=0}^{q-1} Q_\ell(x) \prod_{k \neq \ell} (x - x_{k,t}).
\]

Proof: The result in (56) follows from Theorem 2.1. \( \square \)

In developing the barycentric formula we distinguish between the cases in which \( p \) is even and odd.

Theorem 5.1 Let \( p \) be even, and for \( \ell = 0, 1, \ldots, q - 1 \), let \( \tilde{Q}_\ell(x) \), be the polynomial of degree \( 2n - 1 \) that satisfies the interpolation conditions
\[
\tilde{Q}_\ell(x_{k,r}) = f_{k+\ell} w_{k,r}, \quad k = 1, l', \quad r = 0, 1, \ldots, n - 1,
\]
Then \( P(x) \) of Lemma 5.1 can be expressed in the form
\[
P(x) = \frac{\sum_{\ell=0}^{q-1} \frac{(-1)^{q-\ell}}{(2\pi)^2} \sin 2\pi \theta \tilde{Q}_\ell(x)/(T_{2n}(x) - T_{2n}(x_\ell))}{\sum_{\ell=0}^{q-1} \frac{(-1)^{q-\ell}}{(2\pi)^2} \sin 2\pi \theta /((T_{2n}(x) - T_{2n}(x_\ell))},
\]
where \( T_{2n}(x) \) is the Chebyshev polynomial of the first kind.

Proof: For \( \ell = 0, 1, \ldots, q - 1 \), we let \( R_\ell(x) \) be the polynomial of degree \( 2n - 1 \) that satisfies the interpolation conditions
\[
R_\ell(x_{k,r}) = w_{k,r}, \quad k = 1, l', \quad r = 0, 1, \ldots, n - 1,
\]
on the subset of points \( Y_i \). We then obtain the barycentric formula for \( P(x) \)

\[
(60) \quad P(x) = \frac{\sum_{i=0}^{n-1} Q_i(x) / \prod_{j \neq i} (x - x_{ir}) (x - x_{ir})}{\sum_{i=1}^{n-1} R_i(x) / \prod_{j \neq i} (x - x_{ir}) (x - x_{ir})}
\]

from Theorem 2.2. Observing that

\[
(61) \quad \prod_{k=0}^{r-1} (x - x_{kr}) = 2^{2n+1} (T_{2n}(x) - T_{2n}(x_{l}))
\]

we obtain

\[
(62) \quad \prod_{k \neq l'}^{r} (x_{l'r} - x_{kr}) = \prod_{k \neq l'}^{r} (x_{l'r} - x_{kr}) / \prod_{t=0}^{r-1} (x_{l'r} - x_{kt}) \prod_{t=0}^{r-1} (x_{l'r} - x_{tr})
\]

\[
= \left( \frac{N \sin N\theta_{lr}}{2^{n-1} \sin \theta_{lr}} \right) / \left( 2^{n-1} \sin \theta_{lr} \right)
\]

and hence,

\[
(63) \quad w_{kr} = c_0 (-1)^{l} \sin 2n\theta_{l}, \quad k = l, l', \quad r = 0, 1, \ldots, n - 1
\]

where \( c_0 = 2^{N-2n+1}/p \) is a constant independent of \( l \) and \( r \). Comparing (55),(57) and (59) with (63) we obtain

\[
(64) \quad Q_l(x) = c_0 (-1)^{l} \sin 2n\theta_{l} \hat{Q}_l(x),
\]

\[
(65) \quad R_l(x) = c_0 (-1)^{l} \sin 2n\theta_{l},
\]

for \( l = 0, 1, \ldots, q-1 \). Combining (64),(65) and (61) in (60), we obtain (58). \( \square \)

**Theorem 5.2** Let \( p \) be odd, and for \( l = 0, 1, \ldots, q - 1 \), let \( \hat{Q}_l(x) \), be the polynomial of degree \( 2n - 1 \) that satisfies the interpolation conditions

\[
(66) \quad \hat{Q}_l(x_{kr}) = \begin{cases} (-1)^{l} f_{l+rr}, & k = l' \\ (-1)^{l} f_{l+rr}, & k = l \end{cases}, \quad r = 0, 1, \ldots, n - 1
\]

Furthermore, let \( \hat{Q}_{q-1}(x) \), be the polynomial of degree \( n - 1 \) that satisfies the interpolation conditions

\[
(67) \quad \hat{Q}_{q-1}(x) = f_{q-1+rr}, \quad r = 0, 1, \ldots, n - 1.
\]
Then $P(x)$ of Lemma 5.1 can be expressed in the form

$$
P(x) = \sum_{\ell=0}^{q-2} (-1)^\ell \sin 2\theta_{1,\ell} \hat{Q}_{1,\ell}(x)/(T_n(x) - T_{2\ell}(x)) + \frac{1}{2}(-1)^{q-1} \hat{Q}_{q-1}(x)/T_n(x) - \sum_{\ell=0}^{q-2} (-1)^\ell 2 \sin n\theta_{1,\ell} T_n(x)/(T_{2\ell}(x) - T_{2\ell+1}(x)) + H^{-1}(q-1)/T_n(x) - H^{-1}.$$  

(68)

**Proof:** We start by observing from (62) that for $\ell = 0,1,\ldots,q-2$

$$w_{l,r} = c_0(-1)^{r+1} \sin 2\theta_{l,r},$$  

(69)

$$w_{l,r} = c_0(-1)^{r+1} \sin 2\theta_{l,r}.$$  

(70)

Furthermore,

$$\prod_{t=0}^{n-1} (x - z_{q-1,t}) = 2^{-n+1} T_n(x),$$  

(71)

and therefore,

$$\prod_{k=0}^{q-1, r} (a_{q-1,r} - z_{q-1,r}) = \prod_{k,r \neq q-1} (a_{q-1,r} - z_{q-1,r})/\prod_{t \neq r} (z_{q-1,t} - z_{q-1,t})$$  

$$= \left( \frac{N \sin N \theta_{q-1,r}}{2^{N-1} \sin \theta_{q-1,r}} \right) / \left( \frac{n \sin \theta_{q-1,r}}{\sin \theta_{q-1,r}} \right).$$  

(72)

and finally

$$w_{q-1,r} = c_1(-1)^{r+1},$$

(73)

where $c_1 = c_0 2^{n-1}$ is a constant independent of $q-1$ and $r$. Comparing (55) and (59) with (66),(69) and (70) we obtain for $l = 0,1,\ldots,q-2$,

$$Q_l(x) = c_0(-1)^{l} \sin 2\theta_{l} \hat{Q}_{1,\ell}(x),$$  

(74)

$$R_l(x) = c_0(-1)^{l} \sin 2\theta_{l} T_n(x)/T_n(x).$$  

(75)

Similarly, from (67) and (73) we obtain

$$Q_{q-1}(x) = c_0(-1)^{q-1} \hat{Q}_{q-1}(x),$$  

(76)

$$R_{q-1}(x) = c_0(-1)^{q-1}.$$  

(77)

Combining (74),(75),(76) and (77) in (60), we obtain (68). \(\square\)
We next show how to find the corresponding polynomials $Q_l(z), l = 0, 1, \ldots, q - 1$ using the FFT algorithm. We give explicit formulas for the case where $p$ is odd. The case where $p$ is even is solved similarly. Let $Q(z)$ be a polynomial of degree $m - 1$. Then, $Q(z)$ has a unique representation in terms of the Chebyshev polynomials of order less than $m$, i.e.,

$$Q(z) = \frac{1}{2}a_0 + \sum_{k=1}^{m-1} a_k T_k(z).$$

Rewriting the series in terms of $z = \cos \theta, z = e^{i\theta}$, we get the corresponding complex Chebyshev polynomial of degree $m - 1$

$$C(z) = \sum_{k=-m+1}^{m-1} c_k z^k, \quad c_k = c_{-k} = \frac{1}{2}a_k, \quad k = 0, 1, \ldots, m - 1.$$

Let $Q(z)$ satisfies the interpolation conditions

$$Q(x_j) = g_j, \quad x_j = \cos \theta_j, \quad \theta_j = \frac{2j + 1}{2m}, \quad j = 0, 1, \ldots, m - 1$$

then $C(z)$ satisfies the interpolation conditions

$$C(v_j) = P(x_j) = g_j, \quad v_j = e^{i\theta_j} \quad j = 0, 1, \ldots, m - 1,$$

and vice versa. Hence, $Q(z)$ can be obtained from $C(z)$.

**Theorem 5.3** Let $\hat{C}(z)$ be the balanced complex trigonometric polynomial of degree $n$ that satisfies the interpolation conditions

$$\hat{C}(z) = \frac{1}{2}f_{-i+j} + \sum_{j'=n-1-j}^{n-1} f_{-i+j'} \quad j' = n - 1 - j, \quad j = 0, 1, \ldots, n - 1,$$

where $z_1 = e^{i\pi/2n}$. Then $C(z)$, the complex Chebyshev polynomial of degree $n - 1$ corresponding to $\hat{Q}_{n-1}(z)$ in (68), can be expressed in the form

$$C(z) = \hat{C}(z/z_1^{1/2}) + \hat{C}(1/(z_1^{1/2}))$$
5 CHEBYSHEV INTERPOLATION

Proof: First, it is clear from (83) that $O(z)$ is a complex Chebyshev polynomial. Furthermore, from (23) $O(s)$ is of the form

$$O(z) = \frac{1}{2} c_n z^{-n} + \sum_{k=0}^{n-1} c_k z^k + \frac{1}{2} c_n z^n, \quad c_n = c_n.$$  \hfill (84)

Consequently, the coefficients of $z^n$ and $z^{-n}$ in (83) are

$$\frac{1}{2} c_n (z_1^{-n/2} + z_1^{-n/2}) = 0,$$

and $O(z)$ is of degree at most $n - 1$. Next, $Y_{n-1} = X_{n-1} = \{ \cos \frac{2k+1}{2n} \pi, j = 0, 1, \ldots, n - 1 \}$ is a set of $n$ Chebyshev points, and therefore from (82)

$$C(zj) = O(z_1^j) = \sum_{k=0}^{n} c_k z_1^{k-1} = f_{n-1+j}, \quad j = 0, 1, \ldots, n - 1,$$

and $O(z)$ satisfies the interpolation conditions.  

Theorem 5.4 Let $O(z)$ be the balanced complex trigonometric polynomial of degree $n$ that satisfies the interpolation conditions

$$O(zj) = f(j), \quad j = 0, 1, \ldots, n - 1,$$

where $z_j = e^{i2\pi j/n}$ and $i < q - 1$. Then $O(z)$, the complex Chebyshev polynomial of degree $2n - 1$ corresponding to $(-2i \sin 2\theta Q_n(x))$ in (68), can be expressed in the form

$$O(z) = \left( ( \frac{z}{z_j} )^n - ( \frac{z}{z_j} )^{-n} \right) O(z_j) + \left( ( \frac{1}{z_j} )^n - ( \frac{1}{z_j} )^{-n} \right) O\left( \frac{1}{z_j} \right) \equiv \tilde{O}(z) + \tilde{O}(z^{-1}).$$  \hfill (87)

Proof: First, it is clear from (88) that $O(z)$ is a complex Chebyshev polynomial. Furthermore, from (23) $O(z_j)$ is of the form

$$\tilde{O}(z_j) = \frac{1}{2} c_n (z_j^{-n} + \sum_{k=0}^{n-1} c_k (z_j)^k + \frac{1}{2} c_n (z_j)^n, \quad c_n = c_n.$$  \hfill (88)
Consequently the coefficients of $z^n$ and $z^{-2n}$ in (88) are

\[
\frac{1}{2} c_n ((v_i v_i) - (v_i v_i)^n) = 0,
\]

and $C(z)$ is of degree at most $2n-1$. Next, we observe that

\[
\dot{C}(z) = 2i \sin \theta - \theta_i \dot{C}(\frac{z}{v_i})
\]

and therefore from (87) and (66)

\[
\begin{align*}
C(v_i, j) &= (-1)^{j-1} 2i \sin 2n \theta_i \dot{C}(\frac{z_i}{v_i}) \\
C(v_i, j) &= (-1)^j 2i \sin 2n \theta_i \dot{C}(\frac{z_i^{2n-1-j}}{v_i}),
\end{align*}
\]

and $C(z)$ satisfies the interpolation conditions.

We have the following operation count for constructing and evaluating the polynomial, when $N \sim n(2p), p \ll N$,

<table>
<thead>
<tr>
<th>Operation</th>
<th>Sequential</th>
<th>Parallel</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construction</td>
<td>$N \log N$</td>
<td>$2n \log(2n)$</td>
<td>$\sim p$</td>
</tr>
<tr>
<td>Evaluation</td>
<td>$N$</td>
<td>$2(n + p)$</td>
<td>$\sim p$</td>
</tr>
</tbody>
</table>

Again we obtain a speed-up of order $p$ both for the construction and evaluation of the polynomial as compared to the sequential FFT algorithm.

6 The general Hermite interpolation problem

Let $X = \{z_0, z_1, \ldots, z_N\}$ be given distinct points in the interval $[a, b]$, and let $f(x)$ be a function defined on $[a, b]$, for which

\[
f_j \equiv f^{(j)}(z_j), \quad t = 0, 1, \ldots, k_j - 1, \quad j = 0, 1, \ldots, M,
\]

are given. We are interested in constructing a representation of the general Hermite interpolation polynomial $P(x)$ of degree at most $N$, $N+1 = \sum_{j=0}^{M} k_j$, that interpolates $f(x)$ on $X$, i.e.,

\[
P^{(j)}(z_j) = f_j, \quad t = 0, 1, \ldots, k_j - 1, \quad j = 0, 1, \ldots, M.
\]
and is most suitable for parallel computation. Let \( \{X_1, X_2, \ldots, X_p\} \) be a partition of \( X \), i.e.,
\[
X = \bigcup_{i}^{p} X_i \quad \text{and} \quad X_i \cap X_j = \emptyset \quad \text{for} \quad i \neq j.
\]
The following theorem indicates how \( P(x) \) can be constructed independently and in parallel by \( p \) processors, each solving a smaller general Hermite interpolation problem on one of the subsets \( X_i \).

**Theorem 6.1** For \( i = 1, \ldots, p \), and \( x_j \in X_i \), define
\[
\begin{align*}
\psi_{i,j}^0 &= \prod_{x_r \notin X_i} (x_j - x_r)^{k_r}, \\
z_{i,j}^t &= (-1)^t \sum_{x_r \in X_i} \frac{k_r}{(x_j - x_r)^{t+1}}, \quad t = 0, 1, \ldots, k_j - 2, \\
\psi_{i,j}^t &= \frac{1}{t!} \sum_{s=0}^{t-1} \psi_{i,j}^s \frac{z_{i,j}^{t-s}}{s!}, \quad t = 1, \ldots, k_j - 1, \\
q_{i,j}^t &= \left( \frac{1}{t!} - \sum_{s=0}^{t-1} q_{i,j}^s \psi_{i,j}^{t-s} / \psi_{i,j}^s \right), \quad t = 0, 1, \ldots, k_j - 1,
\end{align*}
\]
and let \( Q_i(x) \) be the polynomial of degree at most \( n_i - 1 \), \( n_i = \sum_{s \in X_i} k_j \), that satisfies the following interpolation conditions:
\[
Q_i^{(t)}(x_j) = t! q_{i,j}^t, \quad t = 0, 1, \ldots, k_j - 1.
\]
Then \( P(x) \), the interpolation polynomial on \( X \), is given by
\[
P(x) = \sum_{i=1}^{p} Q_i(x) \prod_{x_r \notin X_i} (x - x_r)^{k_r} \equiv \sum_{i=1}^{p} Q_i(x) l_i(x).
\]

**Proof:** First, it is clear that the right-hand side of (102) is a sum of polynomials each of degree at most \( N \). Next, we observe that
\[
l_i(x) = l_i(x) \sum_{x_r \notin X_i} \frac{k_r}{x - x_r} \equiv l_i(x) z_{i}(x)
\]
Comparing (98) with (103), and (97),(99) with (105), we see that
\[ (105) \quad z_i^{(t)}(x_j) = tl_i^{t} \], \quad t = 0, 1, \ldots, k_j - 2, \]
and
\[ (106) \quad z_i^{(t)}(x_j) = tl_i^{t} \], \quad t = 0, 1, \ldots, k_j - 1. \]
Differentiating both sides of (102) \( t \) times, and setting \( z = x_j \) in there, we finally obtain
\[ (107) \quad P^{(t)}(x_j) = \sum_{s=0}^{t} \binom{t}{s} Q_i^{(s)}(x_j) I_i^{(t-s)}(x_j) \]
\[ (108) \quad = t! \sum_{s=0}^{t} I_i^{(s)} Q_i^{(t-s)} = f_i^{(t)} \],
as required. The result now follows from the uniqueness of \( P(z) \).

We conclude with the barycentric formula for \( P(z) \) given in Theorem 6.2 below.

**Theorem 6.2** The general Hermite interpolation polynomial \( P(z) \) of Theorem 6.1 has the barycentric form
\[ P(z) = \frac{\sum_{i=1}^{p} Q_i(z) / \prod_{z \in X_i} (z - z_j)^{k_j}}{\sum_{i=1}^{p} R_i(z) / \prod_{z \in X_i} (z - z_j)^{k_j}} \]
where \( R_i(z) \), like \( Q_i(z) \), is a polynomial of degree at most \( n_i - 1 \) that satisfies the same interpolation conditions with \( f_j \) replaced by 1, and \( f_i^{(t)} \) by \( 0 \), \( t = 1, \ldots, k_j - 1 \), for all \( j \).
7 CONCLUSION

Proof: As in Theorem 2.2. □

For example, for the classical Hermite interpolation problem, in which \( k_j = 2 \) for all \( j \), we get

\[
Q_i(x_j) = f_j^i / v_{i,j}^k \quad Q_i'(x_j) = f_j^i / v_{i,j}^0 - f_j \sum_{x \in X; \ x_j = x_r} \frac{\lambda_r}{x_j - x_r},
\]

and

\[
R_i(x_j) = 1 / v_{i,j}^0, \quad R_i'(x_j) = -\sum_{x \in X; \ x_j = x_r} \frac{\lambda_r}{x_j - x_r}.
\]

We can find the formulas for \( v_{i,j}^s, \ s = 0, 1, \ldots, k_j - 1, \) in

\[
O(\sum_{x \in X; \ x_j = x_r} k_j |X - X_r| + k_j^2) \leq O(n_k |X - X_r| + n_k^2)
\]

operations, and the formulas for the \( q_{i,j}^s, \ s = 0, 1, \ldots, k_j, \) in

\[
O(\sum_{x \in X; \ x_j = x_r} k_j^2) \leq O(n_k^3)
\]

operations. Let \( n_i \sim n \sim N/p, i = 1, \ldots, p, \) and assume that the \( v_{i,j}^s \) are known. Each processor is then faced with a general Hermite interpolation problem of order \( n \), that can be solved in \( O(n^3) \) operations.

7 Conclusion

We have presented a new interpolation polynomial that is especially useful for parallel computers as its construction and evaluation requires almost no communication between the processors. The interpolation formula adjusts itself according to the number of processors available, and it receives the form of Lagrange and Newton's formulas in its extreme cases. We further give a generalization to the barycentric formula of Lagrange which is most useful numerically and computationally. The interpolation problem is divided into smaller independent subproblems, which can be solved independently by each processor using any known sequential interpolation method. We have shown that in most cases of interest the formulation of the interpolation subproblems can be done analytically, reducing the problem from order \( N \) to order \( n \).
REFERENCES

\[ n \sim \frac{N}{p} \] and achieving optimum speed-ups. Furthermore, the barycentric formula developed in the present work can be seen to enjoy a high degree of numerical stability as in the case with the barycentric formula for the ordinary Lagrange interpolation.

Practically, small interpolation problems are solved most efficiently using a single sequential processor. Such processors are usually equipped with a mathematical coprocessor implementing in hardware systolic and other parallel algorithms for the basic arithmetic operations. However, for large interpolation problems, the performance will still deteriorate with increasing problem size. We have shown in this work how to gain from both levels of parallelization, i.e., the hardware chip level and the software MIMD level, by splitting the larger problem into smaller subproblems. The preferable size of each interpolation subproblem will depend on the type and performance of each processor in the system.

References


REFERENCES
