Parallel Hermite Interpolation:

An Algebraic Approach

Omer Egecioglu [†], Santa Barbara, **E. Gallopoulos** [‡], Urbana, **Cetin K. Koc** [*], Houston.

Abstract - Zusammenfassung

Parallel Hermite Interpolation: An Algebraic Approach : Given $n+1$ distinct points and arbitrary order derivative information at these points, a parallel algorithm to compute the coefficients of the corresponding Hermite interpolating polynomial in $O(\log n)$ parallel arithmetic operations using $O(n^2)$ processors is presented. The algorithm relies on a novel closed formula that yields the expansion of the generalized divided differences in terms of the given function and derivative values. We show that each one of the coefficients in this expansion and the required linear combinations can be evaluated efficiently.

The particular cases where up to first and second order derivative information is available are treated in detail. The proof of the general case, where arbitrarily high order derivative information is available, involves algebraic arguments that make use of the theory of symmetric functions.

AMS Subject Classifications (1985) : 65D05, 65W05, 68Q25.

Key words : Hermite interpolation, parallel algorithm, parallel prefix, symmetric function.

Parallele Hermitesche Interpolation: Ein algebraischer Zugang : Gegeben seien $n+1$ verschiedene Punkte sowie die Werte von Ableitungen beliebiger Ordnung in diesen Punkten. Fur die Berechnung der Koeffizienten des zugehorigen Hermiteschen Interpolationspolynoms wird ein paralleler Algorithmus vorgestellt, der $O(\log n)$ parallele arithmetische Operationen auf $O(n^2)$ Prozessoren benotigt. Der Algorithmus basiert auf einer neuartigen geschlossenen Darstellung der verallgemeinerten Differenzenquotienten durch die gegebenen Funktions- und Ableitungswerte. Wir zeigen, daβ sowohl die Koeffizienten in dieser Darstellung als auch die benotigten Linearkombinationen effizient berechnet werden konnen.

Detailliert behandelt werden die Spezialfalle, daβ die Ableitungen bis zur ersten bzw. zweiten Ordnung bekannt sind. Fur den Beweis des allgemeinen Falles, wo Ableitungswerte beliebiger hoherer Ordnung verfugbar sind, wird ein algebraischer Zugang gewahlt, bei dem die Theorie symmetrischer Funktionen herangezogen wird.

This article was presented at the First International Conference on Industrial and Applied Mathematics (ICIAM87), Paris, France, June 29 - July 3, 1987.

† Supported in part by NSF Grant No. DCR-8603722.

‡ Supported in part by the National Science Foundation under Grants Nos. NSF DCR84-10110 and NSF DCR85-09770, the U. S. Department of Energy under Grant No. DOE DE-FG02-85ER25001, the Air Force Office of Scientific Research Grant No. AFOSR-85-0211 and an IBM Donation.

* Supported by Lawrence Livermore National Laboratory Contract No. LLNL-7526225.

1. Introduction

Given a collection of $n+1$ pairs $(x_i\ ,\ f_i) \in F \times F$ ($i = 0\ ,\ 1\ ,\ \ldots\ ,\ n$; x_i 's distinct), the *interpolation problem* over F is to construct a polynomial $p_n(x) \in F[x]$ of degree n such that $p_n(x_i) = f_i$ $(i = 0\ ,\ 1\ ,\ \ldots\ ,\ n)$. If $f_i = f(\ x_i\ )$ are the values of a function f at the points x_i , then the polynomial $p_n(x)$ is said to interpolate f at the nodes $x_0\ ,\ x_1\ ,\ \ldots\ ,\ x_n$.

If $p_n(x)$ is expressed in the Newton form

$$p_n(x) = f_0 + f_{01}(x - x_0) + f_{012}(x - x_0)(x - x_1) +$$

$$f_{0123}(x - x_0)(x - x_1)(x - x_2) + \cdots +$$

$$f_{012..n}(x - x_0)(x - x_1)(x - x_2)\cdots(x - x_{n-1})\quad ,$$

then the coefficients $f_{012..p}$; ($p = 0\ ,\ 1\ ,\ \ldots\ ,\ n$) are called the *divided differences* (DD's) of f , which can be computed using Neville's or Aitken's recursion formulae [12] , [5]. As an example the Neville procedure uses the recursion

$$f_{i\ ,\ i+1\ ,\ ..\ ,\ i+q} = \frac{f_{i\ ,\ i+1\ ,\ ..\ ,\ i+q-1} - f_{i+1\ ,\ i+2\ ,\ ..\ ,\ i+q}}{x_i - x_{i+q}}$$

to calculate the terms in the following triangular table:

$$\begin{array}{llll} f_0 & & & \\ f_1 & f_{01} & & \\ f_2 & f_{12} & f_{012} & \\ f_3 & f_{23} & f_{123} & f_{0123} \end{array}$$

The terms on the diagonal are the DD's and hence the coefficients of the Newton polynomial. The Neville and the Aitken procedures require $O(\ n^2\ )$ arithmetic operations. Note that the entries in a given column can be calculated independently of one another, and they depend only on the entries in the previous column and the x_i 's. This gives a straightforward parallel algorithm for the DD's, particularly suitable for systolic implementation (see [4] and [10]), where each column is computed in $O(\ 1\ )$ time using as many processors as there are entries in that particular column. Since the maximum length of a column is n and there are n columns to calculated, this approach requires $O(\ n\ )$ parallel arithmetic operations to calculate all the DD's using $O(\ n\ )$ processors.

A parallel algorithm to calculate the DD's in $O(\log n)$ time using $O(\ n^2\ )$ processors is reported in [3]. Here we will sketch this parallel Newton interpolation algorithm as it is relevant to our treatment of Hermite interpolation.

Setting

$$y_{ij} = \frac{1}{x_i - x_j} \qquad (1.1)$$

for $i \neq j$, the p^{th} divided difference of f can be expressed as a linear combination of the given function values f_0, f_1, $\cdots$, f_n with coefficients that are products of the y_{ij}'s. This expansion is of the form

$$f_{012..p} = (y_{01}\, y_{02} .. y_{0p})\, f_0 + (y_{10}\, y_{12} .. y_{1p})\, f_1 + \cdots + (y_{p0}\, y_{p1} .. y_{p,p-1})\, f_p \quad , \tag{1.2}$$

where p ranges from 0 to n [1]. It will be useful to denote the coefficient of f_i in the linear expansion of $f_{012..p}$ $(i \leq p)$ by $f_{012..p}\,\big|_{f_i}$. Then the coefficients in the expansion (1.2) can be written concisely as

$$f_{012..p}\,\big|_{f_i} = y_{i0}\, y_{i1} \cdots y_{i,i-1}\, y_{i,i+1} \cdots y_{ip} \quad . \tag{1.3}$$

If the coefficients $f_{012..p}\,\big|_{f_i}$ are known for $0 \leq i \leq p \leq n$, then all of the divided differences $(f_0, f_{01}, f_{012}, \cdots, f_{012..n})$ of f that are required for the interpolating polynomial $p_n(x)$ can be calculated in $O(\log n)$ time using $O(n^2)$ processors. This is because for each p, the right hand side of (1.2) can be calculated using $O(\log n)$ parallel arithmetic operations with $O(n)$ processors, and n independent instances of this computation is required for p ranging from 1 to n. The coefficients in (1.3) themselves can also be calculated in $O(\log n)$ time using $O(n^2)$ processors. To see this note that

$$f_{01}\,\big|_{f_0} = y_{01}$$

$$f_{012}\,\big|_{f_0} = y_{01}\, y_{02}$$

$$f_{0123}\,\big|_{f_0} = y_{01}\, y_{02}\, y_{03}$$

$$\cdots$$

$$f_{012..n}\,\big|_{f_0} = y_{01}\, y_{02}\, y_{03} \cdots y_{0n}$$

and therefore the computation of this sequence of coefficients amounts to the calculation of the prefixes of the quantities $(y_{01}, y_{02}, \cdots, y_{0n})$. This can be done by using the parallel prefix algorithm in $\log n$ time using n processors [6], [8]. Since $n+1$ concurrent instances of a parallel prefix algorithm are needed to compute the prefixes of the terms $(y_{i0}, y_{i1}, \cdots, y_{i,i-1}, y_{i,i+1}, \cdots, y_{in})$ for i ranging from 0 to n, the total number of processors required becomes $O(n^2)$. A detailed analysis of this approach for the computation of DD's for the construction of the Newton interpolating polynomial can be found in [3].

The parallel Newton interpolation algorithm is numerically better conditioned than the interpolation algorithms that rely on FFT (such as given in [2] and [7]) regardless of the parallelism involved [3]. Particularly the parallel Newton interpolation algorithm is numerically superior to the parallel algorithm proposed in [11], which constructs the Lagrange interpolating polynomial in $O(\log n)$ time with $O(n^2)$ processors by extensively using the FFT.

In this paper we construct a parallel algorithm for Hermite interpolation. The algorithm computes the coefficients of the Hermite interpolating polynomial in $O(\log n)$ parallel arithmetic steps using

$O(n^2)$ processors for a fixed number of derivatives by making extensive use of parallel prefix algorithms.

The error analysis of the algorithm we present is similar to the analysis of the parallel Newton interpolation algorithm given in [3] and will not be addressed here.

2. Hermite Interpolation

In the most general case of Hermite interpolation, we are given the derivative information

$$f(x_i)\ ,\ f^{(1)}(x_i)\ ,\ \cdots\ ,\ f^{(m_i-1)}(x_i)$$

at $n+1$ distinct points $x_0\ ,\ x_1\ ,\ \ldots\ ,\ x_n$.

Denote $f(x_i)$ by f_i and $\dfrac{f^{(k-1)}(x_i)}{(k-1)!}$ by f_{i^k} . Then the Hermite interpolating polynomial can be expressed in the form

$$P(x) = f_0 + f_{0^2}(x-x_0) + f_{0^3}(x-x_0)^2 + \cdots + f_{0^{m_0}}(x-x_0)^{m_0-1} +$$

$$f_{0^{m_0}1}(x-x_0)^{m_0} + f_{0^{m_0}1^2}(x-x_0)^{m_0}(x-x_1) + \cdots + f_{0^{m_0}1^{m_1}}(x-x_0)^{m_0}(x-x_1)^{m_1-1} +$$

$$f_{0^{m_0}1^{m_1}2}(x-x_0)^{m_0}(x-x_1)^{m_1} + f_{0^{m_0}1^{m_1}2^2}(x-x_0)^{m_0}(x-x_1)^{m_1}(x-x_2) + \cdots + \cdots +$$

$$f_{0^{m_0}1^{m_1}2^{m_2}\ldots n^{m_n}}(x-x_0)^{m_0}(x-x_1)^{m_1}(x-x_2)^{m_2}\cdots(x-x_n)^{m_n-1} \quad .$$

The coefficients $f_{0^{a_0}1^{a_1}\ldots n^{a_n}}$ for various $a_i \le m_i$ are called the *generalized divided differences* (GDD's) of f . The GDD's can be calculated in the same way as the DD's by using the Neville or the Aitken recursion formulae. For instance, the Neville recursion in this case takes the form

$$f_{i^{a_i}\ldots j^{a_j}} = \frac{f_{i^{a_i}\ldots j^{a_j-1}} - f_{i^{a_i-1}\ldots j^{a_j}}}{x_i - x_j}$$

$$= y_{ij}\ (\ f_{i^{a_i}\ldots j^{a_j-1}} - f_{i^{a_i-1}\ldots j^{a_j}}\ )\ , \qquad (2.1)$$

where the terms of the form f_{i^k} with $k \le m_i$ are to be interpreted as $\dfrac{1}{(k-1)!}\ f^{(k-1)}(x_i)$. The correctness of this process can be verified by noting that

$$\frac{d^{k-1}}{dx^{k-1}}\ P(x_i) = (k-1)!\, f_{i^k} = f^{(k-1)}(x_i) \quad 1 \le k \le m_i$$

holds. As an example, given two points x_0 and x_1 with $m_0 = 3$ and $m_1 = 2$, the Neville procedure computes the entries in the following triangle

$$
\begin{array}{lllll}
f_0 & & & & \\
f_0 & f_{0^2} & & & \\
f_0 & f_{0^2} & f_{0^3} & & \\
f_1 & f_{01} & f_{0^2 1} & f_{0^3 1} & \\
f_1 & f_{1^2} & f_{01^2} & f_{0^2 1^2} & f_{0^3 1^2}
\end{array}
$$

using the recursion in (2.1) . As in the case for calculating the DD's, the Neville procedure requires $O(n^2)$ sequential time to calculate all GDD's for fixed values of m_i , $0 \le i \le n$.

Since we are interested in developing a parallel algorithm to calculate all GDD's, we will seek a linear expansion formula of the form (1.2) for $f_{0^{a_0}1^{a_1}\cdots n^{a_n}}$ in terms of f_{j^i} with $0 \le j \le n$ and $1 \le i \le a_i$ for various $a_i \le m_i$. It turns out that the coefficient $f_{0^{a_0}1^{a_1}\cdots n^{a_n}}\big|_{f_{j^i}}$ can be expressed in a closed form, which reduces to the expressions for the coefficients that appear in the Newton polynomial (1.2) when $a_0 = a_1 = \cdots = a_n = 1$.

In most practical instances where Hermite interpolating polynomials are required for the data

(I) $f(x_i)$ and $f'(x_i)$ for $i = 0, 1, \cdots, n$ are given, or

(II) $f(x_i)$, $f'(x_i)$ and $f''(x_i)$ for $i = 0, 1, \cdots, n$ are given,

the algorithm turns out to have an especially simple structure. In Section 4, we describe parallel Hermite interpolation algorithms for these specific cases in detail.

3. Linear Expansion of GDD's

In this section, for brevity of notation we will denote y_{0i} by y_i and represent the GDD $f_{0^r 1^s}$ simply by the string $0^r 1^s$ whenever necessary. The repeated application of (2.1) with two given points x_0 and x_1 can be represented as a signed and weighted binary tree, where the weight associated with each node at level p is y_1^p . The leftson of each node is obtained by dropping a 1 , and the rightson by dropping a 0. The leaves correspond to strings consisting of 0's or 1's only. All the right branches carry a negative sign and the left branches a positive sign, in accordance with the signs produced by repeated application of (2.1) . The sign of a given node is defined to be the product of all the signs on the path from the node to the root. As an example, when $r = 3$ and $s = 2$, we have the following representation of the expansion of f_{00011}

file figure1

Figure 1

From this representation, we see that $f_{00011}\big|_{f_0}$ is equal to the signed sum of all the weights of the leaves labeled 0 . Since each 0 omitted on the path from the root to a given node introduces a negative sign, the sign of each leaf labeled 0 is positive. This gives

$$f_{00011}\,\big|_{f_0} = +3\,y_1^4 \quad .$$

In general, the sign of each leaf labeled 0 in the expansion of $f_{0^r\,1^s}$ will be $(-1)^{r-1}$. Note that all these leaves are at level $r+s-1$. Furthermore, in the expansion

$$f_{0^r\,1^s}\,\big|_{f_0} = (-1)^{r-1}\,C_o\,y_1^{r+s-1} \quad ,$$

the coefficient C_o is the number of leaves labeled 0.

Next, we count the number of leaves labeled 0 in such a tree in the general case. It is not difficult to see that C_o is the number of ways of parenthesizing the string $0^r\,1^s$ starting from $0^{r-1}\,(01)\,1^{s-1}$ in such a way that each new pair of parentheses introduced contains one more symbol than the previous. For example, with this coding, the leftmost leaf in Figure 1 corresponds to the parenthesization $((0\,(0\,(01)))\,1)$, and the rightmost one to $(0\,(0\,((01)\,1)))$.

Let $a_{r,s}$ denote the number of such parenthesizations of the string $0^r\,1^s$. Using this interpretation, (or proceeding directly from the recursive structure of a binary tree) we obtain the recursion

$$a_{r,s} = a_{r-1,s} + a_{r,s-1} \tag{3.1}$$

with $a_{r,1} = a_{1,s} = 1$. By a simple induction, this gives

$$a_{r,s} = \binom{r+s-2}{s-1} \quad .$$

Thus

$$f_{0^r\,1^s}\,\big|_{f_0} = (-1)^{r-1}\binom{r+s-2}{s-1}y_1^{r+s-1} \quad . \tag{3.2}$$

Note that by treating the string 00 as a single symbol the derivation of (3.2) also yields that the sign of each leaf labeled 00 is $(-1)^{r-2}$ and that there are $\binom{r+s-3}{s-1}$ of these. Thus

$$f_{0^r\,1^s}\,\big|_{f_{00}} = (-1)^{r-2}\binom{r+s-3}{s-1}y_1^{r+s-2}$$

and in general we have for $i \le r$

$$f_{0^r\,1^s}\,\big|_{f_{0^i}} = (-1)^{r-i}\binom{r+s-i-1}{s-1}y_1^{r+s-i} \quad . \tag{3.3}$$

The computation of the coefficients of f_1 and f_{11} can be carried out exactly as above by a symmetry argument, giving

$$f_{00011}\,\big|_{f_1} = -3\,y_1^4 \;, \quad f_{00011}\,\big|_{f_{11}} = -y_1^3 \quad .$$

Combining all these observations, f_{00011} has the expansion

$$f_{00011} = 3\,y_1^4\ f_0 - 2\,y_1^3\ f_{00} + y_1^2\ f_{000} - 3\,y_1^4\ f_1 - y_1^3\ f_{11} \quad . \tag{3.4}$$

Equivalently, using the notation introduced in (1.1), the expansion in (3.4) takes the form

$$f_{00011} = (3\,y_{01}^4)\ f_0 + (-2\,y_{01}^3)\ f_{00} + (y_{01}^2)\ f_{000} + (-3\,y_{10}^4)\ f_1 + (y_{10}^3)\ f_{11} \quad .$$

Now we turn to the general case of computing the coefficients f_{0^i} in the expansion of $f_{0^{a_0}1^{a_1}\cdots n^{a_n}}$. Even though the combinatorial treatment of the derivation of (3.3) can be extended to this case, we will proceed by induction. First, we give a closed formula for the coefficient of f_0 in the expansion of $f_{0^{a_0}1^{a_1}\cdots n^{a_n}}$. Recall that a *composition* or an *ordered partition* of a nonnegative integer m into n parts is a representation of the form

$$m = \lambda_1 + \lambda_2 + \cdots + \lambda_n$$

in which each λ_i is a nonnegative integer and the order of the summands is important. For example, there are exactly four ordered partitions of 3 into 2 parts: $0 + 3$, $1 + 2$, $2 + 1$, and $3 + 0$.

Theorem I

$$f_{0^{a_0}1^{a_1}\cdots n^{a_n}}\ \Big|_{f_0} = (-1)^{a_0-1} \sum_{\lambda_1+\lambda_2+\cdots+\lambda_n\,=\,a_0-1} \binom{\lambda_1+a_1-1}{\lambda_1}\binom{\lambda_2+a_2-1}{\lambda_2}\cdots\binom{\lambda_n+a_n-1}{\lambda_n} y_1^{\lambda_1+a_1}\,y_2^{\lambda_2+a_2}\cdots y_n^{\lambda_n+a_n}$$

where the summation is over all ordered partitions of $a_0 - 1$ *into* n *parts.*

Proof

Note first of all that $a_0 = r$ and $a_1 = s$ gives

$$f_{0^r\,1^s}\ \Big|_{f_0} = (-1)^{r-1} \sum_{\lambda_1\,=\,r-1} \binom{\lambda_1+s-1}{\lambda_1} y_1^{\lambda_1+s} = (-1)^{r-1}\binom{r+s-2}{s-1} y_1^{r+s-1} \quad ,$$

in agreement with (3.2). Furthermore, in the special case where $a_0 = a_1 = \cdots = a_n = 1$, we obtain the coefficient of the Newton interpolating polynomial

$$f_{01\ldots n}\ \Big|_{f_0} = y_1\,y_2\cdots y_n \quad ,$$

as given in (1.3).

For convenience, set

$$C_{\mathbf{a},\,m} = \sum_{\lambda_1+\lambda_2+\cdots+\lambda_n\,=\,m} \binom{\lambda_1+a_1-1}{\lambda_1}\binom{\lambda_2+a_2-1}{\lambda_2}\cdots\binom{\lambda_n+a_n-1}{\lambda_n} y_1^{\lambda_1+a_1}\,y_2^{\lambda_2+a_2}\cdots y_n^{\lambda_n+a_n}$$

for $\mathbf{a} = (a_1\,,\,a_2\,,\,\ldots\,,\,a_n)$ and $m \ge 0$, where the summation is over all ordered partitions of m into n parts. Thus we claim that

$$f_{0^{a_0}1^{a_1}\cdots n^{a_n}}\ \Big|_{f_0} = (-1)^{a_0-1}\ C_{\mathbf{a},\,a_0-1} \quad .$$

First, we construct the generating function $F_{\mathbf{a}}$ of the sequence of numbers $C_{\mathbf{a},\,m}$. Note that by Newton's theorem we have for every i the formal power series expansion

$$\frac{1}{(1-y_i)^{a_i}} \;=\; \sum_{\lambda_i \ge 0} \binom{\lambda_i + a_i - 1}{\lambda_i} y_i^{\lambda_i} \quad , \tag{3.5}$$

so that multiplying these expansions for $i = 1\,,\,2\,,\,\ldots\,,\,n$ gives

$$\frac{y_1^{a_1}\, y_2^{a_2} \cdots y_n^{a_n}}{(1-y_1)^{a_1}\,(1-y_2)^{a_2} \cdots (1-y_n)^{a_n}} = \sum_{\lambda_1\,,\,\lambda_2\,,\,\ldots\,,\,\lambda_n \ge 0} \binom{\lambda_1 + a_1 - 1}{\lambda_1} \binom{\lambda_2 + a_2 - 1}{\lambda_2} \cdots \binom{\lambda_n + a_n - 1}{\lambda_n} y_1^{\lambda_1 + a_1}\, y_2^{\lambda_2 + a_2} \cdots y_n^{\lambda_n + a_n} \quad .$$

It follows that

$$F_{\mathbf{a}} = \sum_{m \ge 0} C_{\mathbf{a}\,,\,m}\; t^m = \frac{y_1^{a_1}\, y_2^{a_2} \cdots y_n^{a_n}}{(1 - t\,y_1)^{a_1}\,(1 - t\,y_2)^{a_2} \cdots (1 - t\,y_n)^{a_n}} \quad . \tag{3.6}$$

From the recursive formula for the GDD's given in (2.1) , we have

$$f_{0^{a_0}1^{a_1}\cdots n^{a_n}} = y_n \left(f_{0^{a_0}1^{a_1}\cdots n^{a_n - 1}} \;-\; f_{0^{a_0 - 1}1^{a_1}\cdots n^{a_n}} \right) \quad .$$

Therefore,

$$f_{0^{a_0}1^{a_1}\cdots n^{a_n}} \,\Big|_{f_0} = y_n \; f_{0^{a_0}1^{a_1}\cdots n^{a_n - 1}} \,\Big|_{f_0} \;-\; y_n \; f_{0^{a_0 - 1}1^{a_1}\cdots n^{a_n}} \,\Big|_{f_0} \quad . \tag{3.7}$$

By induction on $\sum_{i=0}^{n} a_i$, we may assume that the two coefficients on the right hand side of (3.7) are given by

$$y_n\,(-1)^{a_0 - 1}\; F_{(a_1\,,\,a_2\,,\,\ldots\,,\,a_n - 1)} \,\Big|_{t^{a_0 - 1}} \qquad \text{and} \qquad y_n\,(-1)^{a_0 - 2}\; F_{(a_1\,,\,a_2\,,\,\ldots\,,\,a_n)} \,\Big|_{t^{a_0 - 2}} \quad ,$$

respectively. Thus, to prove the theorem, it suffices to show that

$$F_{(a_1\,,\,a_2\,,\,\ldots\,,\,a_n)} \,\Big|_{t^{a_0 - 1}} = y_n\, F_{(a_1\,,\,a_2\,,\,\ldots\,,\,a_n - 1)} \,\Big|_{t^{a_0 - 1}} \;+\; y_n\, F_{(a_1\,,\,a_2\,,\,\ldots\,,\,a_n)} \,\Big|_{t^{a_0 - 2}}$$

$$= y_n\, F_{(a_1\,,\,a_2\,,\,\ldots\,,\,a_n - 1)} \,\Big|_{t^{a_0 - 1}} \;+\; y_n\, t\, F_{(a_1\,,\,a_2\,,\,\ldots\,,\,a_n)} \,\Big|_{t^{a_0 - 1}} \quad .$$

But by (3.6) , this reduces to the verification of the functional identity

$$\frac{y_1^{a_1}\, y_2^{a_2} \cdots y_n^{a_n}}{(1 - t\,y_1)^{a_1}\,(1 - t\,y_2)^{a_2} \cdots (1 - t\,y_n)^{a_n}} \;= \tag{3.8}$$

$$= y_n\, \frac{y_1^{a_1}\, y_2^{a_2} \cdots y_n^{a_n - 1}}{(1 - t\,y_1)^{a_1}\,(1 - t\,y_2)^{a_2} \cdots (1 - t\,y_n)^{a_n - 1}} \;+\; y_n\, t\; \frac{y_1^{a_1}\, y_2^{a_2} \cdots y_n^{a_n}}{(1 - t\,y_1)^{a_1}\,(1 - t\,y_2)^{a_2} \cdots (1 - t\,y_n)^{a_n}} \quad ,$$

which is immediate. □

The general formula for the coefficient $f_{0^{a_0}1^{a_1}\cdots n^{a_n}}\,\big|_{f_{0^i}}$ can be stated as follows:

Theorem II

$$f_{0^{a_0}1^{a_1}\cdots n^{a_n}}\,\Big|_{f_{0^i}} = (-1)^{a_0 - i} \sum_{\lambda_1+\lambda_2+\cdots+\lambda_n = a_0 - i} \binom{\lambda_1 + a_1 - 1}{\lambda_1}\binom{\lambda_2 + a_2 - 1}{\lambda_2}\cdots\binom{\lambda_n + a_n - 1}{\lambda_n} y_1^{\lambda_1+a_1}\, y_2^{\lambda_2+a_2}\cdots y_n^{\lambda_n+a_n}$$

for every $i \le a_0$ *, where the summation is over all ordered partitions of* $a_0 - i$ *into* n *parts.*

Proof

The proof is similar to the proof of Theorem I and will be omitted. □

Note that by using the notation for the y_{ij}'s given in (1.1) the formula for $f_{0^{a_o}1^{a_1}\cdots n^{a_n}}\,\big|_{f_{ji}}$ in the general case takes the form

$$(-1)^{a_j - i} \sum \binom{\lambda_o + a_o - 1}{\lambda_o}\cdots\binom{\lambda_{j-1} + a_{j-1} - 1}{\lambda_{j-1}}\binom{\lambda_{j+1} + a_{j+1} - 1}{\lambda_{j+1}}\cdots\binom{\lambda_n + a_n - 1}{\lambda_n} y_{j0}^{\lambda_o+a_o}\cdots y_{j,\,j-1}^{\lambda_{j-1}+a_{j-1}}\, y_{j,\,j+1}^{\lambda_{j+1}+a_{j+1}}\cdots y_{jn}^{\lambda_n+a_n},$$

where the summation is carried over all ordered partitions $\lambda_o + \cdots + \lambda_{j-1} + \lambda_{j+1} + \cdots + \lambda_n$ of $a_j - i$.

4. Special Hermite Interpolating Polynomials

In this section we will give algorithms for the parallel computation of the GDD's for the following important cases:

CASE (I) $f(x_i)$ and $f'(x_i)$ given for $0 \le i \le n$, and

CASE (II) $f(x_i)$, $f'(x_i)$ and $f''(x_i)$ given for $0 \le i \le n$.

As we remarked in Section 2, if the coefficients of f_{j^i} are known for all $0 \le j \le n$ and $i \le m_j$ then all the necessary GDD's, namely the terms of the form $f_{0^{a_0}1^{a_1}\,..\,p^{a_p}}$ where $k \ge a_0 \ge a_1 \ge .. \ge a_p$ and $1 \le p \le n$, can be calculated in $O(\log n)$ time using $O(n^2)$ processors for a fixed value of k. Hence the problem reduces to calculating the coefficients

$$f_{0^{a_0}1^{a_1}\,..\,p^{a_p}}\,\Big|_{f_{j^i}}$$

for $1 \le i \le k$ and $0 \le j \le p \le n$. Now we will show that, for $k = 2$ and $k = 3$ corresponding to the cases (1) and (2) above, these coefficients can also be calculated in $O(\log n)$ time with $O(n^2)$ processors.

CASE (1) ($k = 2$)

From Theorem I we have

$$f_{0^2 1^{a_1}\,\cdots\,p^{a_p}}\,\Big|_{f_0} = (-1) \sum_{\lambda_1+\lambda_2+\cdots+\lambda_p = 1} \binom{\lambda_1 + a_1 - 1}{\lambda_1}\binom{\lambda_2 + a_2 - 1}{\lambda_2}\cdots\binom{\lambda_p + a_p - 1}{\lambda_p} y_1^{\lambda_1+a_1}\, y_2^{\lambda_2+a_2}\cdots y_p^{\lambda_p+a_p}$$

for $1 \le p \le n$. This equation simplifies to

$$f_{0^2 1^{a_1} \cdots p^{a_p}} \Big|_{f_0} = - y_1^{a_1} y_2^{a_2} \cdots y_p^{a_p} \sum_{i=1}^{p} a_i y_i \quad .$$

Similarly, for the coefficient of f_{0^2} , we obtain from Theorem II that

$$f_{0^2 1^{a_1} \cdots p^{a_p}} \Big|_{f_{0^2}} = y_1^{a_1} y_2^{a_2} \cdots y_p^{a_p} \quad .$$

Since $2 \geq a_1 \geq a_2 \geq \cdots \geq a_n \geq 1$, as p ranges from 1 to n we obtain the following table for the coefficients of f_{0^2} :

$$f_{0^2 1} \Big|_{f_{0^2}} = y_1$$

$$f_{0^2 1^2} \Big|_{f_{0^2}} = y_1^2$$

$$f_{0^2 1^2 2} \Big|_{f_{0^2}} = y_1^2 y_2$$

$$f_{0^2 1^2 2^2} \Big|_{f_{0^2}} = y_1^2 y_2^2$$

$$f_{0^2 1^2 2^2 3} \Big|_{f_{0^2}} = y_1^2 y_2^2 y_3$$

$$f_{0^2 1^2 2^2 3^2} \Big|_{f_{0^2}} = y_1^2 y_2^2 y_3^2$$

$$\cdots$$

$$f_{0^2 1^2 2^2 3^2 \cdots n} \Big|_{f_{0^2}} = y_1^2 y_2^2 y_3^2 \cdots y_n$$

$$f_{0^2 1^2 2^2 3^2 \cdots n^2} \Big|_{f_{0^2}} = y_1^2 y_2^2 y_3^2 \cdots y_n^2 \quad .$$

Clearly these are the prefixes of the quantities $(y_1 , y_1 , y_2 , y_2 , y_3 , y_3 , \cdots , y_n , y_n)$, and hence they can be calculated in $\log 2n$ time using $2n$ processors. Due to symmetry of the GDD's, all of the terms

$$f_{0^2 1^2 2^2 \cdots (p-1)^2 p} \Big|_{f_{j^2}} \quad , \quad f_{0^2 1^2 2^2 \cdots (p-1)^2 p^2} \Big|_{f_{j^2}} \qquad 0 \leq j \leq p \quad , \quad 1 \leq p \leq n$$

can be calculated using $n+1$ instances of the parallel prefix algorithm. Hence $O(n^2)$ processors suffice to calculate all of them in $O(\log n)$ time.

For the terms $f_{0^2 1^2 2^2 \cdots (p-1)^2 p} \Big|_{f_0}$ and $f_{0^2 1^2 2^2 \cdots (p-1)^2 p^2} \Big|_{f_0}$ we obtain the following formulae :

$$f_{0^2 1} \Big|_{f_0} = - y_1 \, (y_1)$$

$$f_{0^2 1^2} \Big|_{f_0} = - {y_1}^2 \, (2 y_1)$$

$$f_{0^2 1^2 2^2}\,\big|_{f_0} = -{y_1}^2\, y_2\,(2y_1 + y_2)$$

$$f_{0^2 1^2 2^2}\,\big|_{f_0} = -{y_1}^2\, {y_2}^2\,(2y_1 + 2y_2)$$

$$f_{0^2 1^2 2^2 3}\,\big|_{f_0} = -{y_1}^2\, {y_2}^2\, y_3\,(2y_1 + 2y_2 + y_3)$$

$$f_{0^2 1^2 2^2 3^2}\,\big|_{f_0} = -{y_1}^2\, {y_2}^2\, {y_3}^2\,(2y_1 + 2y_2 + 2y_3)$$

$$\cdots$$

$$f_{0^2 1^2 2^2 3^2 \cdots n}\,\big|_{f_0} = -{y_1}^2\, {y_2}^2\, {y_3}^2 \cdots y_n\,(2y_1 + 2y_2 + 2y_3 + \cdots + y_n)$$

$$f_{0^2 1^2 2^2 3^2 \cdots n^2}\,\big|_{f_0} = -{y_1}^2\, {y_2}^2\, {y_3}^2 \cdots {y_n}^2\,(2y_1 + 2y_2 + 2y_3 + \cdots + 2y_n)\ .$$

The terms outside the parentheses are obtained by negating the coefficients of f_{0^2}, and hence they need not be calculated again. For the terms in the parentheses notice that by applying the parallel prefix algorithm to the quantities $(2y_1, 2y_2, \cdots, 2y_n)$, where the operation is taken to be addition, we get the terms $2y_1$, $2y_1 + 2y_2$, $2y_1 + 2y_2 + 2y_3$, etc. The other half of the terms can be calculated from these by doing only one parallel addition. Hence again $O(n)$ processors suffice to calculate all the coefficients of f_0 in the linear expansion of $f_{0^2 1^{a_1} \cdots p^{a_p}}$ for $2 \geq a_1 \geq \cdots \geq a_p \geq 1$ and $1 \leq p \leq n$. The coefficients of f_j for j ranging from 0 to n are found by $n+1$ concurrent applications of the procedure explained above. Hence the coefficients of all f_j and f_{j^2} in the expansion of $f_{0^2 1^{a_1} 2^{a_2} \cdots p^{a_p}}$ for $1 \leq p \leq n$ and $2 \geq a_1 \geq a_2 \geq \cdots \geq a_n \geq 1$ can be found in $O(\log n)$ time using $O(n^2)$ processors.

CASE (2) ($k = 3$)

In this case we are interested in the coefficients of f_0, f_{0^2} and f_{0^3} in the linear expansion of $f_{0^3 1^{a_1} 2^{a_2} \cdots p^{a_p}}$ for $3 \geq a_1 \geq \cdots \geq a_p \geq 1$. We will start with the simplest one

$$f_{0^3 1^{a_1} 2^{a_2} \cdots p^{a_p}}\,\big|_{f_{0^3}} = {y_1}^{a_1}\, {y_2}^{a_2} \cdots {y_p}^{a_p}\quad ,$$

which yields the following set of coefficients :

$$f_{0^3 1}\,\big|_{f_{0^3}} = y_1$$

$$f_{0^3 1^2}\,\big|_{f_{0^3}} = {y_1}^2$$

$$f_{0^3 1^3}\,\big|_{f_{0^3}} = {y_1}^3$$

$$f_{0^3 1^3 2}\,\big|_{f_{0^3}} = {y_1}^3\, y_2$$

$$f_{0^3 1^3 2^2}\,\big|_{f_{0^3}} = {y_1}^3\,{y_2}^2$$

$$f_{0^3 1^3 2^3}\,\big|_{f_{0^3}} = {y_1}^3\,{y_2}^3$$

$$\cdots$$

$$f_{0^3 1^3 2^3 \cdots n}\,\big|_{f_{0^3}} = {y_1}^3\,{y_2}^3 \cdots y_{n-1}^3\,y_n$$

$$f_{0^3 1^3 2^3 \cdots n^2}\,\big|_{f_{0^3}} = {y_1}^3\,{y_2}^3 \cdots y_{n-1}^3\,{y_n}^2$$

$$f_{0^3 1^3 2^3 \cdots n^3}\,\big|_{f_{0^3}} = {y_1}^3\,{y_2}^3 \cdots y_{n-1}^3\,{y_n}^3 \quad .$$

It is clear that these quantities can be calculated in $\log 3n$ time using $3n$ processors. By applying $n+1$ concurrent instances of the parallel prefix algorithm we find all $f_{0^3 1^{a_1} \cdots p^{a_p}}\,\big|_{f_{j^3}}$ in $O(\log n)$ time using $O(n^2)$ processors for $3 \geq a_1 \geq \geq a_2 \geq \cdots \geq a_p \geq 1$ and $1 \leq p \leq n$.

For the coefficient of f_{0^2} in the expansion of $f_{0^3 1^{a_1} \cdots p^{a_p}}$, from Theorem II we obtain the following formula,

$$f_{0^2 1^{a_1} \cdots p^{a_p}}\,\big|_{f_{0^2}} = -\,y_1^{a_1}\,y_2^{a_2} \cdots y_p^{a_p} \sum_{i=1}^{p} a_i\,y_i \quad ,$$

which can be given explicitly as

$$f_{0^3 1}\,\big|_{f_{0^2}} = -\,y_1\,(\,y_1\,)$$

$$f_{0^3 1^2}\,\big|_{f_{0^2}} = -\,{y_1}^2\,(\,2\,y_1\,)$$

$$f_{0^3 1^3}\,\big|_{f_{0^2}} = -\,{y_1}^3\,(\,3\,y_1\,)$$

$$f_{0^3 1^3 2}\,\big|_{f_{0^2}} = -\,{y_1}^3\,y_2\,(\,3\,y_1 + y_2\,)$$

$$f_{0^3 1^3 2^2}\,\big|_{f_{0^2}} = -\,{y_1}^3\,{y_2}^2\,(\,3\,y_1 + 2\,y_2\,)$$

$$f_{0^3 1^3 2^3}\,\big|_{f_{0^2}} = -\,{y_1}^3\,{y_2}^3\,(\,3\,y_1 + 3\,y_2\,)$$

$$\cdots$$

$$f_{0^3 1^3 2^3 \cdots n}\,\big|_{f_{0^2}} = -\,{y_1}^3\,{y_2}^3 \cdots y_{n-1}^3\,y_n\,(\,3\,y_1 + 3\,y_2 + \cdots + y_n\,)$$

$$f_{0^3 1^3 2^3 \cdots n^2}\,\big|_{f_{0^2}} = -\,{y_1}^3\,{y_2}^3 \cdots y_{n-1}^3\,{y_n}^2\,(\,3y_1 + 3y_2 + \cdots + 2y_n\,)$$

$$f_{0^3 1^3 2^3 \cdots n^3}\,\big|_{f_{0^2}} = -\,{y_1}^3\,{y_2}^3 \cdots y_{n-1}^3\,{y_n}^3\,(\,3y_1 + 3y_2 + \cdots + 3y_n\,) \quad .$$

The terms outside the parentheses are simply obtained by negating the coefficients of f_{0^3} as it is similar to the $k = 2$ case. Using n processors we can calculate $3y_1$, $3y_1 + 3y_2$, ... , $3y_1 + 3y_2 + \cdots + 3y_n$ in $\log n$ time because they are the prefixes of the terms $(\,3y_1\,,\,3y_2\,,\,3y_3\,,\cdots,\,3y_n\,)$. The rest of the terms in parentheses are calculated from these in $O(1)$ time by doing parallel additions using n processors. It follows that $O(n^2)$ processors are sufficient to calculate all of the coefficients $f_{0^3 1^{a_1} 2^{a_2} \cdots p^{a_p}}\,\big|_{f_{j^2}}$ for $3 \ge a_1 \ge a_2 \ge \cdots \ge a_p \ge 1$, $1 \le p \le n$ and $0 \le j \le n$ in $O(\log n)$ time.

To compute the coefficients of f_0 in the expansion of $f_{0^3 1^{a_1} \cdots p^{a_p}}$ for all $1 \le p \le n$ we again use the theorem I.

$$f_{0^3 1^{a_1} \cdots p^{a_p}}\,\big|_{f_0} = \sum_{\lambda_1 + \lambda_2 + \cdots + \lambda_p = 2} \binom{\lambda_1 + a_1 - 1}{\lambda_1} \binom{\lambda_2 + a_2 - 1}{\lambda_2} \cdots \binom{\lambda_p + a_p - 1}{\lambda_p} y_1^{\lambda_1 + a_1}\, y_2^{\lambda_2 + a_2} \cdots y_p^{\lambda_p + a_p}$$

The sum $\lambda_1 + \lambda_2 + \cdots + \lambda_p$ can be equal to 2 only in two ways :

(i) $\lambda_i = 2$ for $1 \le i \le p$,

(ii) $\lambda_i = \lambda_j = 1$ for $1 \le i$, $j \le p$ and $i \ne j$.

By separating these two cases in the sum operation we obtain

$$f_{0^3 1^{a_1} \cdots p^{a_p}}\,\big|_{f_0} = {y_1}^{a_1}\,{y_2}^{a_2} \cdots {y_p}^{a_p} \left(\sum_{1 \le i \le p} \binom{a_i + 1}{2} y_i^2 \;+ \sum_{\substack{1 \le i,\, j \le p \\ i \ne j}} \binom{a_i}{1} \binom{a_j}{1} y_i\, y_j \right)$$

$$= {y_1}^{a_1}\,{y_2}^{a_2} \cdots {y_p}^{a_p} \left(\sum_{1 \le i \le p} \frac{a_i\,(a_i + 1)}{2}\, y_i^2 \;+ \sum_{\substack{1 \le i,\, j \le p \\ i \ne j}} a_i\, a_j\, y_i\, y_j \right)$$

$$= {y_1}^{a_1}\,{y_2}^{a_2} \cdots {y_p}^{a_p} \left(\sum_{1 \le i \le p} \frac{a_i\,(a_i + 1)}{2}\, y_i^2 \;+ \frac{1}{2} \left[\sum_{1 \le i \le p} a_i\, y_i \right]^2 - \frac{1}{2} \left[\sum_{1 \le i \le p} {a_i}^2\, y_i^2 \right] \right)$$

$$= \frac{1}{2}\,{y_1}^{a_1}\,{y_2}^{a_2} \cdots {y_p}^{a_p} \left(\sum_{1 \le i \le p} a_i\, y_i^2 \;+ \left[\sum_{1 \le i \le p} a_i\, y_i \right]^2 \right) \quad .$$

These quantities can be arranged in a table as in the previous cases :

$$f_{0^3 1}\,\big|_{f_0} = \frac{1}{2}\,y_1\,[\,{y_1}^2 + (y_1)^2\,]$$

$$f_{0^3 1^2}\,\big|_{f_0} = \frac{1}{2}\,{y_1}^2\,[\,2{y_1}^2 + (2y_1)^2\,]$$

$$f_{0^3 1^3}\Big|_{f_0} = \frac{1}{2} y_1{}^3 [\, 3 y_1{}^2 + (3 y_1)^2 \,]$$

$$f_{0^3 1^3 2}\Big|_{f_0} = \frac{1}{2} y_1{}^3 y_2 [\, 3 y_1{}^2 + y_2{}^2 + (3 y_1 + y_2)^2 \,]$$

$$f_{0^3 1^3 2^2}\Big|_{f_0} = \frac{1}{2} y_1{}^3 y_2{}^2 [\, 3 y_1{}^2 + 2 y_2{}^2 + (3 y_1 + 2 y_2)^2 \,]$$

$$f_{0^3 1^3 2^3}\Big|_{f_0} = \frac{1}{2} y_1{}^3 y_2{}^3 [\, 3 y_1{}^2 + 3 y_2{}^2 + (3 y_1 + 3 y_2)^2 \,]$$

$$\cdots$$

$$f_{0^3 1^3 2^3 \cdots (n-1)^3 n}\Big|_{f_0} = \frac{1}{2} y_1{}^3 y_2{}^3 \cdots y_{n-1}^3 y_n [\, 3 y_1{}^2 + 3 y_2{}^2 + \cdots + 3 y_{n-1}^2 + y_n^2 + (3 y_1 + 3 y_2 + \cdots + 3 y_{n-1} + y_n)^2 \,]$$

$$f_{0^3 1^3 2^3 \cdots (n-1)^3 n^2}\Big|_{f_0} = \frac{1}{2} y_1{}^3 y_2{}^3 \cdots y_{n-1}^3 y_n{}^2 [\, 3 y_1{}^2 + 3 y_2{}^2 + \cdots + 3 y_{n-1}^2 + 2 y_n{}^2 + (3 y_1 + 3 y_2 + \cdots + 3 y_{n-1} + 2 y_n)^2 \,]$$

$$f_{0^3 1^3 2^3 \cdots (n-1)^3 n^3}\Big|_{f_0} = \frac{1}{2} y_1{}^3 y_2{}^3 \cdots y_{n-1}^3 y_n{}^3 [\, 3 y_1{}^2 + 3 y_2{}^2 + \cdots + 3 y_{n-1}^2 + 3 y_n{}^2 + (3 y_1 + 3 y_2 + \cdots + 3 y_{n-1} + 3 y_n)^2 \,] .$$

It is not difficult to see that all of these coefficients can be calculated in $O(\log n)$ time using only $O(n)$ processors. We conclude that the coefficients of f_0 , f_{0^2} and f_{0^3} can be computed in $O(\log n)$ time using $O(n^2)$ processors. Hence all GDD's of the form $f_{0^3 1^{a_1} \ldots p^{a_p}}$ can be calculated in $O(\log n)$ time using $O(n^2)$ processors for $3 \geq a_1 \geq a_2 \geq \cdots \geq a_p \geq 1$ and $1 \leq p \leq n$.

Explicit processor and arithmetic operation counts for the above algorithms, as well as the serial and parallel complexities of the classical algorithms (Neville and Aitken) for the two special cases I and II covered in this section appear in Table 1.

	Parallel Hermite Interpolation		Parallel Neville/Aitken		Neville/Aitken
	Processors	Time	Processors	Time	Time
CASE I f , f' given	$2n(n+1)$	$3\log n + 5$	$4n$	$3n$	$\frac{9}{2} n(n+1)$
CASE II f , f' , f'' given	$3n(n+1)$	$4\log n + 3 + 4\log 3$	$9n$	$3n$	$\frac{19}{2} n(n+1)$

Table 1

5. The General Case

Next, we will show that in the most general case, where up to k^{th} order derivatives are given (i.e. each $a_i \le k$), the computation of the coefficients of the Hermite interpolating polynomial can still be performed in $O(\log n)$ parallel time using $O(n^2)$ processors.

Theorem III

The coefficients of an arbitrary order Hermite interpolating polynomial can be computed in $O(\log n)$ *parallel time using* $O(n^2)$ *processors.*

Proof

The idea of the proof rests on the theory of symmetric functions, and we give a sketch of the basic ideas involved.

Given a positive integer N and a set of variables $y_1, y_2, \ldots, y_n$ with $n \ge N$, denote by Λ_N the space of symmetric polynomials in these variables homogeneous of total degree N with rational coefficients. The m^{th} homogeneous symmetric function $h_m(y_1, y_2, \ldots, y_n)$ and the m^{th} power symmetric function $\psi_m(y_1, y_2, \ldots, y_n)$ are defined by setting

$$h_m(y_1, y_2, \ldots, y_n) = \sum_{1 \le i_1 \le i_2 \le \cdots \le i_m \le n} y_{i_1} y_{i_2} \cdots y_{i_m} \tag{5.1}$$

$$\psi_m(y_1, y_2, \ldots, y_n) = \sum_{i \ge 1} y_i^m \quad , \tag{5.2}$$

with $h_o = \psi_o = 1$. Furthermore, for any partition λ of N (i.e $\lambda = (\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_N \ge 0)$ with $\sum_{i=1}^{N} \lambda_i = N$), put

$$h_\lambda = h_{\lambda_1} h_{\lambda_2} \cdots h_{\lambda_N} \quad , \qquad \psi_\lambda = \psi_{\lambda_1} \psi_{\lambda_2} \cdots \psi_{\lambda_N} \quad .$$

It is well known [9] that the collections $\{ h_\lambda \}$ and $\{ \psi_\lambda \}$, where λ runs through all partitions of N, form rational bases for Λ_N. Homogeneous and power symmetric functions are related by the determinantal formula

$$m!\, h_m = \det \begin{pmatrix} \psi_1 & -1 & 0 & \cdots & 0 \\ \psi_2 & \psi_1 & -2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \psi_{m-1} & \psi_{m-2} & \cdots & \cdots & -m+1 \\ \psi_m & \psi_{m-1} & \cdots & \cdots & \psi_1 \end{pmatrix} , \tag{5.3}$$

which holds independent of the number of variables. Thus, we have an expansion of the form

$$h_\lambda(y_1, y_2, \cdots, y_n) = \sum_\mu c_\mu \psi_\mu(y_1, y_2, \cdots, y_n) \quad ,$$

for some rational numbers c_μ where μ runs through all partitions of N. Note that by (5.3), the coefficients c_μ are independent of n.

From (5.1), it easily follows that the generating function of the h_m's is given by

$$\sum_{m \geq 0} h_m(y_1, y_2, \cdots, y_n)\, t^m = \frac{1}{(1 - t\,y_1)(1 - t\,y_2)\cdots(1 - t\,y_n)} \quad . \tag{5.4}$$

From the generating function in (5.4), we have that, for any positive integer a,

$$\frac{1}{(1 - t\,y_1)^a (1 - t\,y_2)^a \cdots (1 - t\,y_n)^a} = \sum_{m \geq 0} C_{a,m}(\psi_1, \psi_2, \cdots, \psi_m)\, t^m \tag{5.5}$$

where $C_{a,m}$ is a homogeneous form of degree m.

Now we consider the expansion of the generating function $F_{\mathbf{a}}$ in terms of the power symmetric functions. Note that by the symmetry of the computation of the GDD's we may assume that $a_0 \geq a_1 \geq a_2 \geq \cdots \geq a_n$, and all of these numbers are bounded above by k. The expansion we need for the function $F_{\mathbf{a}}$ is best communicated by an example. Suppose $n = 4$ and $\mathbf{a} = (4, 4, 3, 2)$. Then

$$F_{\mathbf{a}} = \frac{y_1^4 y_2^4 y_3^3 y_4^2}{(1 - t\,y_1)^4 (1 - t\,y_2)^4 (1 - t\,y_3)^3 (1 - t\,y_4)^2}$$

can be written in the form

$$(y_1^4 y_2^4 y_3^3 y_4^2) \times \frac{1}{(1 - t\,y_1)^2 (1 - t\,y_2)^2 (1 - t\,y_3)^2 (1 - t\,y_4)^2} \times \frac{1}{(1 - t\,y_1)(1 - t\,y_2)(1 - t\,y_3)} \times \frac{1}{(1 - t\,y_1)(1 - t\,y_2)} .$$

Clearly, in such an expansion, the number of products of the form

$$\frac{1}{(1 - t\,y_1)^a (1 - t\,y_2)^a \cdots (1 - t\,y_n)^a}$$

depends only on k.

From the expansion in (5.5), it follows that in general

$$\frac{F_{\mathbf{a}}}{y_1^{a_1} y_2^{a_2} \cdots y_n^{a_n}} \Big|_{t^m}$$

is a linear combination of power symmetric functions $\psi_\lambda(y_1, y_2, \cdots, y_i)$ for various $i \leq n$, where λ is a partition of m.

Now in the expansion (5.5), the number of terms of the form ψ_λ that appear as summands of the functions $C_{a,m}$ is independent of n. This means in return that the number of terms of the form ψ_λ that appear in

$$\frac{F_{\mathbf{a}}}{y_1^{a_1} y_2^{a_2} \cdots y_n^{a_n}} \Big|_{t^m}$$

is independent of n. Note further that for each power symmetric function ψ_λ that is such a summand, λ is a partition of m. Since we are interested in the coefficients of t^m in $F_{\mathbf{a}}$ for the values

$m = 0\,,\,1\,,\,\cdots\,,\,a_i - 1$ and $a_i \le k$ for every i, the number of individual power symmetric functions ψ_r that enters as a product in ψ_λ only depends on k and not on n.

It follows that the coefficients of the Hermite interpolating polynomial can be calculated in $O(\log n)$ parallel time using $O(n^2)$ processors, provided that each $\psi_r(y_1\,,\,y_2\,,\,\cdots\,,\,y_i)$ and each fixed product of the form $y_1^{a_1}\,y_2^{a_2}\cdots y_n^{a_n}$ can be computed in $O(\log n)$ time using $O(n^2)$ processors. But for each such computation, the analogue of the parallel prefix algorithm that was demonstrated in the computation of the cases $k = 2$ and $k = 3$ is applicable. Our claim follows from this observation. □

We remark that even though the number of parallel arithmetic steps required to compute the coefficients of the Hermite interpolating polynomial is guaranteed to be logarithmic in n with $O(n^2)$ processors in the general case, the constants hidden in the big O notation are necessarily exponential in k if no other shortcuts are taken into account. This is not surprising in view of the formula for the coefficient of f_0 given in Theorem I, since the summation involved is over all ordered partitions, and there are exponentially many ordered partitions of k into n parts in general. As an example, the expansion of

$$f_{0^3 1^3 2^3 \cdots (n-1)^3 n^2}\,\Big|_{f_0}$$

in terms of the power symmetric functions is obtained by taking the coefficient of t^2 in

$$\frac{y_1^3\,y_2^3\cdots y_{n-1}^3\,y_n^2}{(1-t\,y_1)^3\,(1-t\,y_2)^3\cdots(1-t\,y_{n-1})^3\,(1-t\,y_n)^2} =$$

$$y_1^3\,y_2^3\cdots y_{n-1}^3\,y_n^2\left[\sum_{m\ge 0} h_m(y_1\,,\,y_2\,,\,.\,.\,,\,y_n)\,t^m\right]^2\left[\sum_{m\ge 0} h_m(y_1\,,\,y_2\,,\,.\,.\,,\,y_{n-1})\,t^m\right]$$

and then expressing each homogeneous symmetric function in terms of the power symmetric functions by the determinantal expansion (5.3). For this particular case, we obtain the expression

$$f_{0^3 1^3 2^3 \cdots (n-1)^3 n^2}\,\Big|_{f_0} = y_1^3\,y_2^3\cdots y_{n-1}^3\,y_n^2\,\times$$

$$\left[\psi_1^2(y_1\,,\,.\,.\,,\,y_n) + \psi_2(y_1\,,\,.\,.\,,\,y_n) + \frac{1}{2}\,\psi_2(y_1\,,\,.\,.\,,\,y_{n-1}) + \frac{1}{2}\,\psi_2(y_1\,,\,.\,.\,,\,y_{n-1}) + \psi_1^2(y_1\,,\,.\,.\,,\,y_n) + 2\,\psi_1(y_1\,,\,.\,.\,,\,y_n)\,\psi_1(y_1\,,\,.\,.\,,\,y_{n-1})\right]$$

as opposed to the much simpler expansion obtained in Section 4.

Thus, for particular cases involving small values of k, it seems possible to cut down the constants in question by considering special expansions with smaller number of terms than the above treatment would produce. Therefore, the algorithm implied by the above proof for arbitrary k has more of an existential flavor, and special instances (e.g. $k = 4\,,\,5$) can be made more efficient by judicious grouping of the terms involved in the expansion of the formula in Theorem II.

References

[1] Abramowitz M. and Stegun I.A.: Handbook of Mathematical Functions, pp. 877, Dover, 1972.

[2] Aho A. , Hopcroft J.E., and Ullman J.D.: The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

[3] Egecioglu O., Gallopoulos E., and Koc C.K.: Fast and practical parallel polynomial interpolation, Technical Report No. 646, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, January, 1987, submitted to Journal of Parallel and Distributed Computing.

[4] Koc C.K.: Parallel Algorithms for Interpolation and Approximation, PhD. Thesis, Department of ECE, University of California, Santa Barbara, July 1988.

[5] Krogh F.: Efficient algorithms for polynomial interpolation and divided differences, Mathematics of Computation, Vol. 24, No. 109, pp. 185-190, January 1970.

[6] Kruskal C.P., Rudolph L., and Snir M.: The power of parallel prefix, IEEE Transactions on Computers, Vol. C-34, No. 10, pp. 965-968, October 1985.

[7] Kung H.T.: Fast evaluation and interpolation, Technical Report, Department of Computer Science, Carnegie-Mellon University, January 1973.

[8] Ladner R. and Fischer M.: Parallel prefix computation, Journal of ACM, Vol. 27, No. 4, pp. 831-838, October 1980.

[9] MacDonald I.G: Symmetric Functions and Hall Polynomials, Oxford University Press, London, 1979.

[10] McKeown G.P.: Iterated Interpolation using a Systolic Array, ACM Transactions on Mathematical Software, Vol. 12, No. 2, pp. 162-170, June 1986.

[11] Reif J.: Logarithmic depth circuits for algebraic functions, SIAM Journal on Computing, Vol. 15, No. 1, pp. 231-242, February 1986.

[12] Tsao N.K. and Prior R.: On multipoint numerical interpolation, ACM Transaction on Mathematical Software, Vol. 4, No. 1, pp. 51-56, March 1978.

Omer Egecioglu

Department of Computer Science

University of California Santa Barbara

Santa Barbara, CA 93106 USA

E. Gallopoulos

Center for Supercomputing Research and Development

University of Illinois at Urbana-Champaign

Urbana, IL 61801 USA

Cetin K. Koc

Department of Electrical Engineering

University of Houston

Houston, TX 77204 USA

.

file figure1

Figure 1

.

	Parallel Hermite Interpolation		Parallel Neville/Aitken		Neville/Aitken
	Processors	Time	Processors	Time	Time
CASE I f , f' given	$2\,n(n+1)$	$3\log n+5$	$4\,n$	$3\,n$	$\frac{9}{2}\,n(n+1)$
CASE II f , f' , f'' given	$3\,n(n+1)$	$4\log n+3+4\log 3$	$9\,n$	$3\,n$	$\frac{19}{2}\,n(n+1)$

Table 1