MATHEMATICS OF COMPUTATION
VOLUME 54, NUMBER 189
JANUARY 1990, PAGES 231–243

EVALUATION OF MULTIVARIATE POLYNOMIALS AND THEIR DERIVATIVES

J. CARNICER AND M. GASCA

ABSTRACT. An extension of Horner's algorithm to the evaluation of m-variate polynomials and their derivatives is obtained. The schemes of computation are represented by trees because this type of graph describes exactly in which order the computations must be done. Some examples of algorithms for one and two variables are given.

1. INTRODUCTION

It is well known that Horner's algorithm for evaluation of univariate polynomials and their derivatives is based upon the nested reformulation of the polynomial. Our aim is to extend this idea to multivariate polynomials.

Let $K[X]$ be the ring of m-variate polynomials on a field K, and P be the set of polynomials of exact total degree 1.

Consider $p \in K[X]$, written in the form

$$p = a + \sum_{i=1}^{n} f_i p_i, \tag{1.1}$$

where $n \in N_0 := N \cup \{0\}$, $a \in K$, $f_i \in P$, and $p_i \in K[X]$, $i = 1, \ldots, n$. As usual, empty sums are 0.

Obviously this representation is not unique. We will restrict our attention to the most interesting case where p_i $(i = 1, 2, \ldots, n)$ are of lower degree than p. Observe that this need not always be true because, for example, one could write

$$-2x = 1 + x \cdot (2x^2 + x) + (-2x - 1) \cdot (x^2 + 1).$$

For any linear form L on $K[X]$ one has

$$L(p) = aL(1) + \sum_{i=1}^{n} L(f_i p_i), \tag{1.2}$$

and sometimes $L(f_i p_i)$ can be easily computed from $L(f_i)$ and $L(p_i)$. In this case it is possible to construct an algorithm to compute $L(p)$ recursively. That happens, for example, when L is

$$L(p) := p(u), \qquad u \in K^m, \tag{1.3}$$

Received April 26, 1988; revised January 6, 1989.
1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 65D15, 68R10, 05C05.
Key words and phrases. Evaluation, multivariate polynomials, derivatives.
Both authors were partially supported by C.I.C.Y.T.

since $L(f_i p_i) = L(f_i) \cdot L(p_i)$. As we shall see, Leibniz's rule for derivatives leads to a similar situation when L is the evaluation of a derivative.

2. Multivariate polynomials and trees

Since the form (1.1) of writing polynomials can be graphically described by trees, we recall some related definitions (see [1, 4]).

Definition 2.1. Let I be a nonempty finite set and σ a binary relation in I. The ordered pair (I, σ) is called a directed (labelled) graph, and the elements of I are called labels.

Definition 2.2. Two directed (labelled) graphs (I, σ) and (I', σ') are isomorphic if there is a bijection $\varphi: I \to I'$ such that

$$(\varphi(x), \varphi(y)) \in \sigma' \Leftrightarrow (x, y) \in \sigma. \tag{2.1}$$

We consider two isomorphic directed graphs as identical.

A directed graph is said to be connected when it cannot be obtained as the union of two disjoint graphs.

Definition 2.3. An oriented tree is a connected directed graph with a label $r \in I$, which is called the root of the tree, such that

(1) for each $y \in I \setminus \{r\}$, there exists a unique $x \in I$ with $(y, x) \in \sigma$ (x is called the successor to y, and y a predecessor of x),
(2) there is no $x \in I$ such that $(r, x) \in \sigma$.

The elements of I which have no predecessor are called initial elements.

Remark 2.4. In the following, we will say only the words "graph" and "tree" instead of "directed graph" and "oriented tree".

As a consequence, an ordering is established in I, saying that $x \prec y$ if and only if there exist (unique) $z_1, z_2, \ldots, z_j$ such that $(x, z_1), (z_1, z_2), \ldots, (z_j, y)$ belong to σ. Roughly speaking, y lies on the unique "path" from x to the root.

Let us consider now the set of all multi-indices $\bigcup_{k \geq 0} N^k$. We write 0 for the empty multi-index $0 = ( )$; then one has $N^0 = \{0\}$ and $N^k = \{(i_1, \ldots, i_k) | i_j \in N,\ j = 1, \ldots, k\}$ for all $k \geq 1$. If $i \in N^k$, we say that the length of i is $l(i) = k$, and thus $l(0) = 0$.

Definition 2.5. Let $i = (i_1, i_2, \ldots, i_{l(i)})$, $j = (j_1, j_2, \ldots, j_{l(j)}) \in \bigcup_{k \geq 0} N^k$ be multi-indices. We define the juxtaposition of i and j by

$$ij := (i_1, i_2, \ldots, i_{l(i)}, j_1, j_2, \ldots, j_{l(j)}) \in \bigcup_{k \geq 0} N^k. \tag{2.2}$$

Remark 2.6. In particular, we have for the empty multi-index 0,

$$i0 = 0i = i \quad \forall i.$$

Given any tree, let us take now as the set of its labels a finite set of multi-indices

$$I \subset \bigcup_{k \geq 0} N^k \tag{2.3}$$

of the following form. The root must be 0, and every label is taken so that if $i = (i_1, i_2, \ldots, i_{l(i)})$ is a label with n_i predecessors, these are the labels $i1 = (i_1, i_2, \ldots, i_{l(i)}, 1)$, $i2 = (i_1, i_2, \ldots, i_{l(i)}, 2), \ldots, in_i = (i_1, i_2, \ldots, i_{l(i)}, n_i)$. In the case of the root 0, the predecessors are $1 = (1), 2 = (2), \ldots, n_0 = (n_0)$.

An example of a tree and its labelling is shown in Figure 2.1, where we draw the root at the bottom.

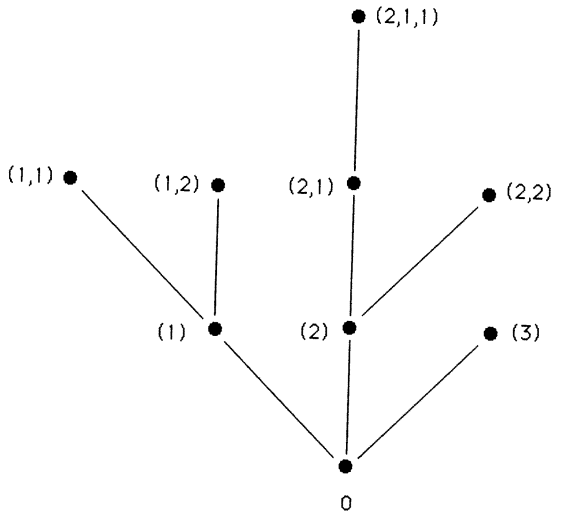

FIGURE 2.1

Given a (labelled) tree (I, σ) and two mappings

$$\begin{aligned} I &\to K, \\ i &\mapsto a_i, \end{aligned} \tag{2.4}$$

$$\begin{aligned} \sigma &\to P, \\ (i, j) &\mapsto f_i, \end{aligned} \tag{2.5}$$

we associate to each label i a polynomial p_i,

$$p_i = a_i + \sum_{h=1}^{n_i} f_{ih} \cdot p_{ih}, \tag{2.6}$$

where n_i is the cardinality of the set of predecessors of i. That is, $p_i = a_i$ if i is an initial label and $p_i = a_i + \sum_{h=1}^{n_i} f_{ih} \cdot p_{ih}$ otherwise. In particular, $p_0 = a_0 + \sum_{h=1}^{n_0} f_h \cdot p_h$ for the root of the tree.

Since we have defined an ordering in I, formula (2.6) gives a recursive way to get the polynomial

$$p_0 = \sum_{i \in I} a_i f_{i_1} f_{i_1 i_2} \cdots f_{i_1 i_2 \cdots i_{l(i)}}, \tag{2.7}$$

where, as usual, the empty product is 1.

Figure 2.2 shows the tree of Figure 2.1 equipped with the two mappings of formulae (2.4) and (2.5).

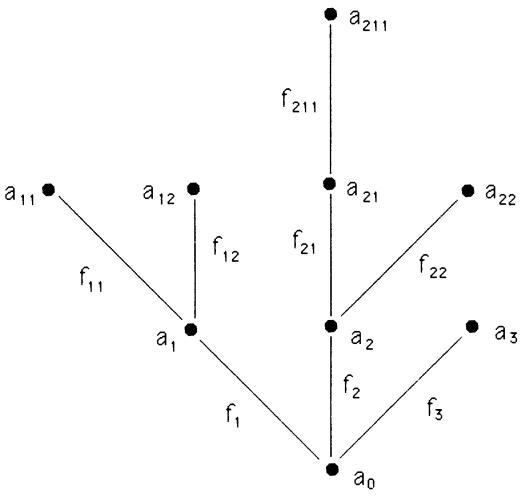

FIGURE 2.2

In this particular case, formula (2.6) gives

$$\begin{aligned}
&p_{211} = a_{211}, \\
&p_{11} = a_{11}, && p_{12} = a_{12}, \\
&p_{21} = a_{21} + f_{211} p_{211}, && p_{22} = a_{22}, \\
&p_1 = a_1 + f_{11} p_{11} + f_{12} p_{12}, && p_2 = a_2 + f_{21} p_{21} + f_{22} p_{22}, \quad p_3 = a_3, \\
&p_0 = a_0 + f_1 p_1 + f_2 p_2 + f_3 p_3.
\end{aligned}$$

3. Evaluation of a polynomial

Let us consider the problem of evaluating at $u \in K^m$ the polynomial p_0 of (2.7) by using successively formulae of type (2.6). Since we have

$$p_0(u) = a_0 + \sum_{h=1}^{n_0} f_h(u) \cdot p_h(u) \tag{3.1}$$

and $f_h(u)$ is easily computed, evaluation of p_0 is reduced to the computation of each $p_h(u)$, $h = 1, \ldots, n_0$. And in order to compute every p_h, we need to know each $p_{hk}(u)$, $k = 1, \ldots, n_h$, and so on:

$$p_i(u) = a_i + \sum_{h=1}^{n_i} f_{ih}(u) \cdot p_{ih}(u), \qquad i \in I. \tag{3.2}$$

This process finishes in a finite number of steps, since we always reach in the end polynomials p_i which are associated with initial labels and are explicitly given by

$$p_i(u) = a_i . \tag{3.3}$$

Formulae (3.1), (3.2), (3.3) really constitute an evaluation algorithm.

Now we must pay attention to a problem of practical implementation. What remains to be solved is the question of the order in which the computations are to be done. The most general answer is: every order of the computational steps is acceptable if the following condition is satisfied:

$$i \prec j \Rightarrow p_i \text{ is evaluated before } p_j . \tag{3.4}$$

In particular, the polynomial p_0 associated with the root is always the last one to be evaluated.

Let $l_{\max}$ be

$$l_{\max} := \max_{i \in I} l(i) . \tag{3.5}$$

We show now an algorithm to evaluate $p_0(u)$ in which the order of the computations is made by a criterion on the lengths of the labels.

Algorithm 3.1.
begin
  for $l = l_{\max}$ **to** 0 **step** -1
    for all $i \in \{i \in I | l(i) = l\}$
      if i is an initial node
        $p_l(u) = a_l$
      else
        $p_l(u) = a_l + \sum_{h=1}^{n_l} f_{lh}(u) \cdot p_{lh}(u)$
      end if
    next i
  next l
end.

A second criterion of ordering for the same computations consists in making reference to the greatest "length" of a path to a label from an initial label. Roughly speaking, we evaluate first every polynomial associated with an initial label, then polynomials associated with labels next to initial labels, etc., until the root is reached. More precisely, we define the set of labels

$$I_l := \left\{ i \in I \,\middle|\, \max_{lh \in I} l(h) = l \right\} \tag{3.6}$$

for $l = 0, \ldots, l_{\max}$. In particular, one has $I_0 = \{i \in I | i \text{ is an initial label}\}$ and $I_{l_{\max}} = \{0\}$, i.e., the root.

Algorithm 3.2.
begin
  for $l = 0$ **to** $l_{\max}$
    for all $i \in I_l$
      if $l = 0$

```
            p_i(u) = a_i
        else
            p_i(u) = a_i + Σ_{h=1}^{n_i} f_{ih}(u) · p_{ih}(u)
        end if
    next i
next l
end.
```

In the example of Figures 2.1 and 2.2 one has $l_{\max} = 3$ and

$$I_0 = \{(1,1),(1,2),(2,1,1),(2,2),(3)\},$$
$$I_1 = \{(2,1),(1)\},$$
$$I_2 = \{(2)\},$$
$$I_3 = \{0\}.$$

Other similar criteria of ordering can be used, as we shall see in the examples of §5.

4. Evaluation of the derivatives of a polynomial

In this section we extend the ideas of §3 to evaluate any derivative of the polynomial p_0 at a point $u \in K^m$. We get an algorithm which includes the evaluation of p_0 as a particular case.

Let $\rho^{(j)} = (\rho_1^{(j)}, \dots, \rho_m^{(j)}) \in K^m$, $j = 1, 2, \dots, n$, be n vectors not necessarily linearly independent. Denote by

$$D_j p = \frac{\partial p}{\partial x_1}\rho_1^{(j)} + \frac{\partial p}{\partial x_2}\rho_2^{(j)} + \cdots + \frac{\partial p}{\partial x_m}\rho_m^{(j)} \tag{4.1}$$

the derivative of p in the direction $\rho^{(j)}$, and by $D_j p(u)$ the value of $D_j p$ at $u \in K^m$

Given any multi-index $t = (t_1, t_2, \dots, t_n) \in N_0^n$, we write

$$|t| = t_1 + t_2 + \cdots + t_n, \tag{4.2}$$

$$D^t = D_1^{t_1} D_2^{t_2} \cdots D_n^{t_n}, \tag{4.3}$$

and

$$E^t = \frac{1}{t!}D^t = \frac{1}{t_1!t_2!\cdots t_n!}D_1^{t_1}D_2^{t_2}\cdots D_n^{t_n}. \tag{4.4}$$

In the following we shall consider two kinds of multi-indices: $t \in N_0^n$ for the upper index of differential operators, and $i \in I \subset \bigcup_{k\geq 0} N^k$ for the labels of a tree. While the ordering of labels i is prescribed by the structure of the tree (see §2), the upper indices t are ordered in the usual way: for $t = (t_1, t_2, \dots, t_n)$, $s = (s_1, s_2, \dots, s_n) \in N_0^n$,

$$t \leq s \Leftrightarrow t_j \leq s_j, \qquad j = 1, \dots, n. \tag{4.5}$$

Let us apply Leibniz's rule

$$E^t(gq)(u) = \sum_{s\leq t} E^{t-s}g(u)\cdot E^s q(u) \quad \forall g, q \in K[X], \ \forall u \in K^m, \ \forall t \in N_0^n \tag{4.6}$$

to formula (2.6). Since f_{ih} are polynomials of degree 1, we get for any t with $|t| \geq 1$,

$$E^t p_i(u) = \sum_{h=1}^{n_i} \left(\sum_{j=1}^{n} D_j f_{ih} \cdot E^{t-e_j} p_{ih}(u) \right) + \sum_{h=1}^{n_i} f_{ih}(u) \cdot E^t p_{ih}(u), \tag{4.7}$$

where e_j is the jth canonical vector $(0, \ldots, 0, 1, 0, \ldots, 0)$ and E^{t-e_j} is taken as the zero operator if $t_j = 0$. Observe that the constants $D_j f_{ih}$ can be easily computed. In the particular case of i being an initial label, one obtains $E^t p_i(u) = 0$.

Formula (4.7) means that $E^t p_i(u)$ can be computed in terms of lower-order derivatives $(E^{t-e_j} p_{ih}(u))$ and derivatives of the same order $(E^t p_{ih}(u))$ applied to polynomials whose labels are the predecessors of i.

In other words, if we denote

$$b_i^t = \sum_{h=1}^{n_i} \left(\sum_{j=1}^{n} D_j f_{ih} \cdot E^{t-e_j} p_{ih}(u) \right) \tag{4.8}$$

(in particular, $E^t p_i(u) = b_i^t = 0$ for initial labels), formula (4.7) reads

$$E^t p_i(u) = b_i^t + \sum_{h=1}^{n_i} f_{ih}(u) \cdot E^t p_{ih}(u), \tag{4.9}$$

which has the same structure as (3.2).

This suggests an algorithm for evaluating derivatives. In order to compute $E^t p_i(u)$, we need the values of

$$E^s p_j(u) \quad \forall j \prec i, \ \forall s \leq t.$$

Hence, similarly to (3.4), the computations can be done in any order which satisfies the following condition:

$$s \leq t, \ i \prec j \Rightarrow E^s p_i \text{ is evaluated before } E^t p_j. \tag{4.10}$$

Observe that by computing $p_0(u)$ by Algorithm 3.1 or 3.2 we obtain as intermediate results $p_i(u)$, all that is needed to compute $E^t p_0(u)$ with $|t| = 1$. In general, having computed $E^{t-e_j} p_0(u)$, $j = 1, \ldots, n$, we have obtained as intermediate results $E^{t-e_j} p_i(u)$, which are needed to compute $E^t p_0(u)$.

As was remarked after (4.8), $E^t p_i(u) = b_i^t = 0$ for all initial labels when $|t| \geq 1$. In general, it is easily seen that

$$E^t p_i(u) = b_i^t = 0 \quad \forall i \in I_0 \cup \cdots \cup I_{|t|-1}. \tag{4.11}$$

Since they are the most typical examples, let us see what happens for

$$E^t = E^{(0, \ldots, 0, k, 0, \ldots, 0)} = \frac{1}{k!} D_j^k \tag{4.12}$$

and for

$$E^t = E^{(1, 1, 0, \ldots, 0)} = D_1 D_2. \tag{4.13}$$

In the first case, for any i,

$$b_i^{(0,\dots,0,k,0,\dots,0)} = \sum_{h=1}^{n_i} D_j f_{ih} \cdot E^{(0,\dots,0,k-1,0,\dots,0)} p_{ih}(u),$$

$$E^{(0,\dots,0,k,0,\dots,0)} p_i(u) = b_i^{(0,\dots,0,k,0,\dots,0)} + \sum_{h=1}^{n_i} f_{ih}(u) \cdot E^{(0,\dots,0,k,0,\dots,0)} p_{ih}(u). \tag{4.14}$$

Then, after computing $E^{(0,\dots,0,k-1,0,\dots,0)} p_i(u)$ for all $i \in I$, we can compute each of the $E^{(0,\dots,0,k,0,\dots,0)} p_i(u)$ using, for example, the ordering in Algorithm 3.1 or 3.2 for the labels i. $E^{(0,\dots,0,k,0,\dots,0)} p_0(u)$ is the last value computed.

For $t = (1, 1, 0, \dots, 0)$ one has

$$b_i^{(1,1,0,\dots,0)} = \sum_{h=1}^{n_i} (D_1 f_{ih} \cdot E^{(0,1,0,\dots,0)} p_{ih}(u)) + \sum_{h=1}^{n_i} (D_2 f_{ih} \cdot E^{(1,0,0,\dots,0)} p_{ih}(u)), \tag{4.15}$$

$$E^{(1,1,0,\dots,0)} p_i(u) = b_i^{(1,1,0,\dots,0)} + \sum_{h=1}^{n_i} f_{ih}(u) \cdot E^{(1,1,0,\dots,0)} p_{ih}(u).$$

In this case we must compute first

$$p_0(u) = E^{(0,\dots,0)} p_0(u),$$

with $p_i(u) = E^{(0,\dots,0)} p_i(u)$ as intermediate results. This allows us to compute $E^{(1,0,0,\dots,0)} p_0(u)$ as well as $E^{(0,1,0,\dots,0)} p_0(u)$. In this process we get $E^{(1,0,0,\dots,0)} p_i(u)$ and $E^{(0,1,0,\dots,0)} p_i(u)$ for all i, and this allows us to compute finally $E^{(1,1,0,\dots,0)} p_0(u)$.

We show now an algorithm to evaluate $E^t p_0(u)$ that also includes the case $t = 0$, which reduces to Algorithm 3.2.

Algorithm 4.1.
begin
  for $k = 0$ **to** $|t|$
    for all s such that $s \le t$, $|s| = k$
      for $l = k$ **to** $l_{\max}$
        for all $i \in I_l$
          if $k = 0$
            $b_l^s = a_i$
          else
            $b_l^s = \sum_{h=1}^{n_l} \left(\sum_{j=1}^{n} D_j f_{ih} \cdot E^{s-e_j} p_{lh}(u) \right)$
          end if
          if $l = k$

$E^s p_i(u) = b_i^s$
else
$E^s p_i(u) = b_i^s + \sum_{h=1}^{n_i} f_{ih}(u) \cdot E^s p_{ih}(u)$
end if
next i
next l
next s
next k
end.

We observe that in the step k only the results obtained in the step $k-1$ must be kept in memory.

Remark 4.2. Given a vector basis $\rho^{(j)}$, $j = 1, 2, \ldots, m$, of the vector space K^m, we can write any derivative in terms of D_j, $j = 1, 2, \ldots, m$. However, derivatives can be computed directly in Algorithm 4.1 for any set of vectors $\{\rho^{(j)} | j = 1, 2, \cdots, n\}$.

5. Some applications: Algorithms for polynomials of one and two variables

We now present some examples of algorithms for the evaluation of polynomials and their derivatives. The notation introduced in §2 has been useful to represent general polynomials. But it can be simplified in many particular cases. This will be shown in each example. For the order of the computational steps, condition (4.10) must be observed, but not necessarily the order of the steps in Algorithm 4.1.

(a) Univariate polynomials. A univariate polynomial, written in the form

$$p_0(x) = \sum_{i=0}^{n} a_i f_0(x) \cdot f_1(x) \cdots f_{i-1}(x) \tag{5.1}$$

(with f_i polynomials of degree one), can be represented as in Figure 5.1.

Algorithm 4.1 with these new notations gives a simple extension of Horner's algorithm to evaluate $p_0^{(t)}(\xi) = D^t p_0(\xi)$, $t \le n$:

Algorithm 5.1
begin
for $k = 0$ **to** t
for $i = n - k$ **to** 0 **step** -1
if $k = 0$
$a_i^{(0)} = a_i$
else
$a_i^{(k)} = Df_i \cdot p_{i+1}^{(k-1)}(\xi)$
end if
if $i = n - k$
$p_{n-k}^{(k)}(\xi) = a_{n-k}^{(k)}$
else
$p_i^{(k)}(\xi) = a_i^{(k)} + f_i(\xi) \cdot p_{i+1}^{(k)}(\xi)$

```
            end if
        next i
    next k
end.
```

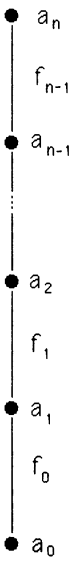

FIGURE 5.1

The most interesting case is when $f_i = x - x_i$, that is

$$p_0 = \sum_{i=0}^{n} a_i (x - x_0)(x - x_1) \cdots (x - x_{i-1}); \tag{5.2}$$

in this case, one has $a_i^{(k)} = p_{i+1}^{(k-1)}(\xi)$ $(k > 0)$ and Algorithm 5.1 can be rewritten as

Algorithm 5.2.

```
begin
    p_n^(0)(ξ) := a_n
    for i = n − 1 to 0 step −1
        p_i^(0)(ξ) = a_i + (ξ − x_i) · p_{i+1}^(0)(ξ)
    next i
    for k = 1 to t
        p_{n−k}^(k)(ξ) = p_{n−k+1}^(k−1)(ξ)
        for i = n − k − 1 to 0 step −1
            p_i^(k)(ξ) = p_{i+1}^(k−1)(ξ) + (ξ − x_i) · p_{i+1}^(k)(ξ)
        next i
    next k
end.
```

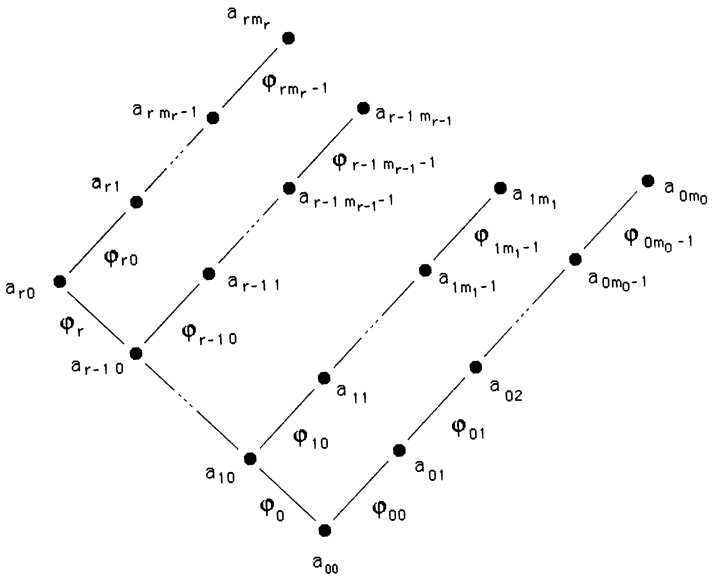

FIGURE 5.2

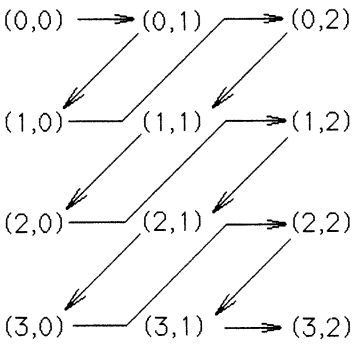

FIGURE 5.3

In Algorithm 5.2, separate steps are made for evaluating the function and for evaluating derivatives, and the values $p_i^{(k)}(\xi)$ are directly computed without using the $a_i^{(k)}$.

In particular, $x_i = 0$, $i = 0, \ldots, n-1$, gives the classical Horner's algorithm.

b) A bivariate example. Consider a bivariate polynomial written in the form

$$(5.3)\quad p_{00}(x, y) = \sum_{i=0}^{r} \sum_{j=0}^{m_i} a_{ij} \varphi_0(x, y) \cdots \varphi_{i-1}(x, y) \varphi_{i0}(x, y) \cdots \varphi_{ij-1}(x, y),$$

where φ_i, φ_{ij} are polynomials of degree 1. This can be represented by a tree as in Figure 5.2.

In order to evaluate

$$p_{00}^{(t,s)}(\xi,\eta) = E^{(t,s)}p_{00}(\xi,\eta) = \frac{1}{t!s!}D_1^t D_2^s p_{00}(\xi,\eta), \qquad t,s \in N_0 \tag{5.4}$$

(where D_1, D_2 are directional derivatives), we first compute all $E^{(v,w)}p_{00}(u)$ with $v \le t$ and $w \le s$. For example, it is possible to do this with the ordering shown in Figure 5.3 (for $(t,s) = (3,2)$).

With this ordering, $p_{00}^{(t,s)} = E^{(t,s)}p_{00}(\xi,\eta)$ can be computed by the following algorithm, where we have used that $\{(v,w)|0 \le v \le t;\ 0 \le w \le s; v+w = k\} = \{(v,x)|\max(0,k-s) \le v \le \min(k,t);\ w = k-v\}$:

Algorithm 5.3.
begin
for $k = 0$ **to** $t+s$
for $v = \max(0,k-s)$ **to** $\min(k,t)$
$w = k - v$
for $i = r$ **to** 0 **step** -1
for $j = m_i$ **to** 0 **step** -1
if $k = 0$
$a_{ij}^{(0,0)} = a_{ij}$
else
if $j = 0$
$a_{i0}^{(v,w)} = D_1\varphi_i \cdot p_{i+10}^{(v-1,w)}(\xi,\eta) + D_2\varphi_i \cdot p_{i+10}^{(v,w-1)}(\xi,\eta)$
$+ D_1\varphi_{i0} \cdot p_{i1}^{(v-1,w)}(\xi,\eta) + D_2\varphi_{i0} \cdot p_{i1}^{(v,w-1)}(\xi,\eta)$
else
$a_{ij}^{(v,w)} = D_1\varphi_{ij} \cdot p_{ij+1}^{(v-1,w)}(\xi,\eta) + D_2\varphi_{ij} \cdot p_{ij+1}^{(v,w-1)}(\xi,\eta)$
end if
end if
if $j = 0$
$p_{i0}^{(v,w)}(\xi,\eta) = a_{i0}^{(v,w)} + \varphi_i(\xi,\eta) \cdot p_{i+10}^{(v,w)}(\xi,\eta)$
$+ \varphi_{i0}(\xi,\eta) \cdot p_{i1}^{(v,w)}(\xi,\eta)$
else
$p_{ij}^{(v,w)}(\xi,\eta) = a_{ij}^{(v,w)} + \varphi_{ij}(\xi,\eta) \cdot p_{ij+1}^{(v,w)}(\xi,\eta)$
end if
next j
next i
next v
next k
end.

In this algorithm, all the terms with a negative upper index, or lower index j greater than m_i, or lower index i greater than r, must be defined to be zero. Further, as a consequence of (4.11), we have

$$a_{ij}^{(v,w)} = 0 \quad \text{for } j > m_i - v - w. \tag{5.5}$$

Similarly to the univariate case, this algorithm gives the general framework for many particular cases, some of them being very simple. For example, when

$$\varphi_i = x - x_i, \quad \varphi_{ij} = y - y_{ij}, \quad D_1 = \frac{\partial}{\partial x}, \quad D_2 = \frac{\partial}{\partial y}, \tag{5.6}$$

we have

$$a_{i0}^{(v,w)} = p_{i+10}^{(v-1,w)}(\xi, \eta) + p_{i1}^{(v,w-1)}(\xi, \eta) \quad \text{for } (v, w) \neq (0, 0), \tag{5.7}$$

$$a_{ij}^{(v,w)} = p_{ij+1}^{(v,w-1)}(\xi, \eta) \quad \text{for } (v, w) \neq (0, 0),\ j \neq 0, \tag{5.8}$$

$$p_{i0}^{(v,w)}(\xi, \eta) = a_{i0}^{(v,w)} + (\xi - x_i) \cdot p_{i+10}^{(v,w)}(\xi, \eta) + (\eta - y_{i0}) \cdot p_{i1}^{(v,w)}(\xi, \eta), \tag{5.9}$$

$$p_{ij}^{(v,w)}(\xi, \eta) = a_{ij}^{(v,w)} + (\eta - y_{ij}) \cdot p_{ij+1}^{(v,w)}(\xi, \eta) \quad \text{for } j \neq 0. \tag{5.10}$$

Algorithm 5.3 can be advantageously used to compute the interpolating polynomial in [2, 3].

Bibliography

1. J. Butcher, *Numerical analysis of differential equations*, Wiley, New York, 1985.
2. M. Gasca and J. I. Maeztu, *On Lagrange and Hermite interpolation in* R^n, Numer. Math. **39** (1982), 1–14.
3. M Gasca and V. Ramirez, *Interpolation systems in* R^k, J. Approx. Theory **42** (1984), 36–51.
4. D. Knuth, *The art of computer programming: Fundamental algorithms*, Addison-Wesley, 1975.
5. L. Schumaker and W. Volk, *Efficient evaluation of multivariate polynomials*, Comput. Aided Geom. Des. **3** (1986), 149–154.

Departamento de Matematica Aplicada, Universidad de Zaragoza, Edificio Matemáticas—Planta 1a, 50009 Zaragoza, Spain