MULTIVARIABLE POLYNOMIAL PROCESSING - APPLICATIONS TO INTERPOLATION

E.V. Krishnamurthy
Department of Applied Mathematics
Indian Institute of Science, Bangalore 560012

H. Venkateswaran
Systems Engineering Division
National Aeronautical Laboratory, Bangalore 560017

Summary

A data-structure suitable for multivariable polynomial processing is introduced. Using this data-structure, arithmetic algorithms are described for addition, subtraction and multiplication of multivariable polynomials; also algorithms are described for forming the inner product and tensor product of vectors, whose components are multivariable polynomials. Application of these algorithms for multivariable cardinal spline approximation is described in detail.

1. Introduction

The processing of multivariable polynomials (m.v.p.) is required for several applications, e.g., computer graphics, topography, System Theory etc. A few important papers on this subject have already appeared.[1-5] In this paper our primary concern is to emphasize the need for the design of a multivariable polynomial processor for applications to interpolation and System Theory. With this in mind we introduce certain special data structures for the representation of the m.v.p. and describe the basic arithmetic algorithms using these data structures.

Also we will describe in detail, how the projection operator technique for multivariable Cardinal Spline interpolation can be implemented using the data structure suggested. Other related applications - such as the computation of g.c.d. of m.v.p., inversion of polynomial matrices, etc., encountered in System Theory - are also indicated.

2. Data Structure

The data structure introduced here is based on viewing a multivariable polynomial (m.v.p.) $P(x_1 .., x_n)$ in the ordered form, $\sum_{i=1}^{m} a_i x_1^{e_{i_1}} x_2^{e_{i_2}} .. x_n^{e_{i_n}}$ (with m terms), where the a_i are elements drawn from a real or finite field. We first define the data structure for a general m.v.p. and then describe the modifications required to use it for Cardinal Spline interpolation applications, where each variable x_i in the m.v.p. can occur in both the powers and plusses form, Sard[6], viz., terms of the form

$$-32\cdot 5(x-3)_+^4\, y^3 z^2$$

where $(x-3)_+ = \begin{cases} x-3 & \text{if } x \geq 3 \\ 0 & \text{if } x < 3 \end{cases}$.

2·1 General multivariable polynomials

A polynomial with two or more variables is represented as two linear sequential lists, one for the degree (DELIST) and the other for the coefficients (CØLIST).

(i) DELIST

Each element in the DELIST consists of degrees of each variable in a term.

(ii) CØLIST

This consists of elements which represent the coefficients of each term in the polynomial; this is set in one to one correspondence with the elements in the DELIST.

Example:

$$P(x,y,z) = 54\cdot 2\, x^4 y^2 z$$

is represented by the ordered structure

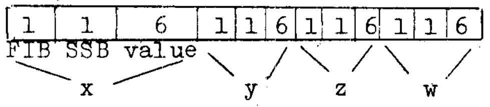
4	2	1	DELIST
54·2			CØLIST

2·2 Interpolating polynomials:

In this case only the DELIST is defined in a different manner. The element in DELIST consists of components corresponding to each variable; a component is subdivided into three parts:

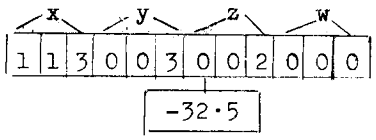
1. Form Indicator Bit (FIB) to indicate conventional or plus (or absolute) form of polynomial.
2. Shift-value Sign Bit (SSB) to indicate the sign of the shift value in the variable in plus (or absolute) form.
3. Value bits - indicating the degree of the variable for conventional form and shift value for plus (or absolute) form.

The bit configurations for 4 ordered variables in a 32-bit word is given below (with one byte (8 bits) for each variable).



Example:

$$-\ 32\cdot5(x+3)_+^4\ \ y^3z^2$$



3. Arithmetic and other algorithms

The arithmetic algorithms to be described below remain the same for both the general m.v.p. and the interpolating polynomials.

3.1 Addition/Subtraction

$(RC,RD,r) \leftarrow (PC,PD,p) \pm (QC,QD,q)$

Let P and Q be two structures with p and q terms respectively. Let R be the resulting structure with r terms.

P=(PC,PD), Q=(QC,QD) and R=(RC,RD)

where PC,QC,RC stand for the coefficient lists and PD,QD,RD for the degree/shift-value (of the plus) structures concerned.

```
Step 0: Set RC(i)=PC(i)
        RD(i)=PD(i), for i=1 to p;
        Set r=p; go to Step 1.
Step 1: Set i=1; go to Step 2.
Step 2: If i>q, go to Step 8; otherwise
        go to Step 3.
Step 3: Set j=1;  go to Step 4.
Step 4: If QD(i)=PD(j) go to Step 6;
        otherwise set j=j+1; go to Step 5
Step 5: If j>p go to Step 7; otherwise
        go to Step 4.
Step 6: RC(j)=RC(j)+QC(i); Set i=i+1;
        go to Step 2.
Step 7: Set r=r+1; RD(r)=QD(i); RC(r)=
        QC(i); Set i=i+1; go to Step 2.
Step 8: The RC list is checked and only
        non-zero coefficient terms are
        retained.  The terms correspond-
        ing to zero coefficients are
        deleted from RD.  Correspondingly
        r is reset.  Stop.
```

Subtraction algorithm steps are the same as addition except Step 6 which becomes,

```
Step 6: RD(j)=RC(j)-QC(i); Set i=i+1;
        go to Step 2.
```

3.2 Multiplication

$(RC,RD,r) \leftarrow (PC,PD,p)x(QC,QD,q)$

Let P and Q be two structures with p and q terms respectively. Let R be the resulting structure with r terms.

P=(PC,PD), Q=(QC,QD) and R=(RC,RD).

Let S=(SC,SD) with s terms be a structure used to hold intermediate results.

```
Step 0: Set RC(i)=PC(i)·QC(1);
        Set RD(i)=PD(i)+QD(1)  for i=1 to
        p;
        Set r=p; Set j=1; Set s=p; go to
        Step 1.
Step 1: Set i=0; Set j=j+1; If j>q go to
        Step 5; otherwise go to Step 2.
Step 2: Set i=i+1; if i>p go to Step 4;
        otherwise go to Step 3.
Step 3: Set SC(i)=PC(i)·QC(j);
        Set SD(i)=PD(i)+QD(j); go to
        Step 2.
Step 4: (RC,RD,r)← ADD (RC,RD,r,SC,SD,s);
        go to Step 1.
Step 5: Stop.
```

3.3 Inner Product

Let $(P_1,P_2,\ldots, P_N)$ and $(Q_1,Q_2,\ldots,Q_N)$ be two sets of N multivariable data structures. Let P_i have p_i terms and Q_i have q_i terms $(i=1,2,\ldots,N)$. Let R be their inner product with r terms.

$P_i=(PC_i,PD_i)$, $Q_i=(QC_i,QD_i)$ and

R=(RC,RD);

Let S=(SC,SD) be a structure used for intermediate storage purposes.

```
Begin:
   $(RC,RD,r) \leftarrow \underline{MUL}(PC1,PD1,P_1,QC1,QD1,q_1)$
 for i=2 to N do
 begin $(SC,SD,s) \leftarrow \underline{MUL}(PC_i,PD_i,p_i,QC_i,QD_i,q_i)$;
      $(RC,RD,r) \leftarrow \underline{ADD}(RC,RD,r,SC,SD,s)$
```

```
        end
End.
```

3·4 Tensor Product

Let $(P_1,P_2,\dots,P_N)$ and $(Q_1,Q_2,\dots,Q_M)$ be two sets of multivariable data structures. Let P_i have p_i terms $(i=1,2,\dots,N)$ and Q_j have q_j terms $(j=1,2,\dots,M)$. Let R_{ij} be their tensor product with r_{ij} terms.

$(i=1,2,\dots,N\ ;\ j=1,2,\dots,M)$

$P_i=(PC_i,PD_i),Q_j=(QC_j,QD_j);R_{ij}=(RC_{ij},RD_{ij})$

```
Begin
    for i=1 to N do
    for j=1 to M do
  $(RC_{ij},RD_{ij},r_{ij})\leftarrow$ MUL$(PC_i,PD_i,p_i,QC_j,QD_j,q_j)$
End
```

Remark: When the tensor product is required recursively, as for example in multivariable polynomial interpolation, the structure for R_{ij} can be modified to be linear (similar to P and Q).

4. Multivariable Interpolation

4·1 Principle of Product operator and Boolean Sum schemes

Our main purpose here is to consider the techniques that are economical using polynomial processing. The well-known method having this feature is the product operator method[7-10] where one breaks up the total interpolation process into n parts, treating the variables $x_1,x_2,\dots,x_n$ separately and using as the basis the functions belonging to the tensor product space.

Let f(x,y) be a two dimensional function to be fitted on the unit square $\{0\leq x\leq 1,\ 0\leq y\leq 1\}$ using the basis function approach. Here we write

$$\Omega\,(x,y) = M_yL_xf(x,y) \qquad (4\cdot1\cdot1)$$

We then find the approximation in two stages; applying an operator L_x to the x variable of f(x,y) and then operator M_y to the y variable. The operator L_x maps on to the space that is spanned by the functions $\phi_i(x)$ $(i=1,2,\dots,m)$ and

$$L_xf(x,y) = \sum_{i=1}^{m} \alpha_i(y)\phi_i(x) \qquad (4\cdot1\cdot2)$$

(More rigorously, one should say that L_x has a range ϕ which is an extended linear space of all polynomials of degree m in x; here the elements of ϕ are of the form $\sum_{i=0}^{m} A_ix^i$, where A_i's are functions of y only. Similarly for M_y.)

The operator M_y maps on to the space that is spanned by the functions $\psi_j(y)$ $(j=1,2,\dots,n)$ and we have an equation of the form,

$$M_yf(x,y) = \sum_{j=1}^{n}\beta_j(x)\ \psi_j(y) \qquad (4\cdot1\cdot3)$$

Thus we get

$$\Omega\ (x,y) = \sum_{i=1}^{m}\ \sum_{j=1}^{n} a_{ij}\phi_i(x)\ \psi_j(y) \qquad (4\cdot1\cdot4)$$

where the product operator maps on to the linear space that is spanned by the product functions of the form

$$\{\phi_i(x)\cdot\psi_j(y);i=1,\dots,m;j=1,\dots,n)\} \qquad (4\cdot1\cdot5)$$

belonging to the tensor product space. If we now consider interpolation points $x_s(s=1,2,\dots,m)$ in [0,1] and the basis functions $\phi_i(x)(i=1,2,\dots,m)$ such that the operator L_x is defined by

$$\begin{aligned}L_xf(x,y) &= \sum_{i=1}^{m}\phi_i(x)\cdot f(x_i,y)\\ &=\{\phi_i(x)\}^t\cdot\{f(x_i,y)\}\ ;\end{aligned} \qquad (4\cdot1\cdot6)$$

here $\{\phi_i(x)\}^t$ indicates the transpose of the column vector of basis functions $\{\phi_i(x)\}$ and the basis functions satisfy the conditions

$$\phi_i(x_s)=\delta_{is}\begin{cases}=0(i\neq s)\\ =1(i=s)\end{cases}\ . \qquad (4\cdot1\cdot7)$$

Considering the interpolation points $y_t(t=1,2,\dots,n)$ in [0,1] and the basis function $\psi_j(y)(j=1,2,\dots,n)$ the operator M_y can be written as

$$\begin{aligned}M_yf(x,y)&=\sum_{j=1}^{n}\psi_j(y)f(x,y_j)\\ &=\{\psi_j(y)\}^t\cdot\{f(x,y_j)\}\ ;\end{aligned} \qquad (4\cdot1\cdot8)$$

again the basis functions satisfy the conditions

$$\psi_j(y_t)=\delta_{jt}\begin{cases}=0(t\neq j)\\=1(t=j)\end{cases} . \qquad (4\cdot1\cdot9)$$

The result of the product operation is then

$$\Omega(x,y)=M_y L_x f(x,y)$$
$$=M_y \sum_{i=1}^{m} \phi_i(x)\cdot f(x_i,y) \qquad (4\cdot1\cdot10)$$
$$=\sum_{i=1}^{m} \sum_{j=1}^{n} \phi_i(x)\psi_j(y) f(x_i,y_j) .$$

Since (4·1·7), (4·1·9) and (4·1·10) give the equation

$$\Omega(x_s,y_t)=f(x_s,y_t),\begin{cases}s=1,2,..,m\\t=1,2,..,n\end{cases} \qquad (4\cdot1\cdot11)$$

we see that applying the operators L_x and M_y separately to f(x,y) is equivalent to choosing the coefficients a_{ij} of the function

$$\Omega(x,y) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}\phi_i(x)\psi_j(y) \qquad (4\cdot1\cdot12)$$

so as to satisfy the interpolation conditions (4·1·11) on the rectangular grid of points $(x_s,y_t),\begin{cases}s=1,2,..,m\\t=1,2,..,n\end{cases}$.

Equation (4·1·10) can be viewed as

$$[\{\phi_i(x)\}^t \otimes \{\psi_j(y)\}^t]\cdot \mathcal{V} f(x_i,y_j) \qquad (4\cdot1\cdot13)$$

where $\otimes$ defines the Krönecker product such that

$$A \otimes B = \{a_{ij}\}\cdot B$$
$$\text{for}\{a_{ij}\} = A;\ \{b_{ij}\} = B \qquad (4\cdot1\cdot14)$$

and $\mathcal{V}(A) = [v\ ]$ transpose of the row vector obtained by placing the rows of A in succession as a long sequence (from left to right and top to bottom order) or

$$v_{n(i-1)+j}=a_{ij}\begin{cases}i=1,2,..,m\\j=1,2,..,n\end{cases} \qquad (4\cdot1\cdot15)$$

Thus to obtain tensor product approximation we need the computation of a tensor product of two row vectors whose elements are polynomials (basis functions) and also the computation of an inner product.

An extension of the product method is due to Gordon[7] In this method, one computes

$$\Omega^+(x,y)=[L_x+M_y-M_yL_x]f(x,y)$$
$$=(L_x\oplus M_y)f(x,y) \qquad (4\cdot1\cdot16)$$

where $\oplus$ is known as the Boolean Sum.

Gordon has shown that under the commutativity assumption, $L_x \oplus M_y$ is algebraically better than M_yL_x, in the sense that the error in the Boolean-Sum scheme is smaller than the tensor product scheme. For specific choices of projectors L_x and M_y this has been shown to be the case and explicit error bounds are available[7]

The function $\phi_i(x)$(as well as $\psi_j(y)$) is also called a blending function since it blends the curves $\{f(x_i,y)\}$ (i=1,..,m) into a surface

$$L_xf(x,y)=\sum_{i=1}^{m} \phi_i(x)f(x_i,y)$$
$$=\{\phi_i(x)\}^t\cdot\{f(x_i,y)\} \qquad (4\cdot1\cdot17)$$

(·defines the inner product of the two vectors).

Surfaces such as L_xf, M_yf and $(L_x \oplus M_y)f$ are referred to as blended surfaces and the operators L_x, M_y and $L_x\oplus M_y$ are called transfinite projectors as their images and f match on more than a finite number of points.

We can now rewrite $\Omega^+(x,y)$ in the form

$$\Omega^+(x,y)=\{\phi_i^t(x)\}\cdot\{f(x_i,y)\}+\{\psi_j^t(y)\}\cdot\{f(x,y_j)\}$$
$$-[\phi^t(x)\otimes\psi^t(y)]\cdot\mathcal{V}(f(x_i,y_j))\ ;$$

where $\{\phi_i(x)\}^t=\phi^t(x);\ \{\psi_j(y)\}^t=\psi^t(y)$ (4·1·18)

In order to gain accuracy Gordon[7] suggests a blending approximation

$$\Omega^*(x,y)=[M_y^*L_x+M_yL_x^*-M_yL_x]f(x,y) \qquad (4\cdot1\cdot19)$$

where $$L_x^*f(x,y)=\{\phi_i^*(x)\}^t\cdot\{f(x_i^*,y)\} \qquad (4\cdot1\cdot20)$$

$$M_y^*f(x,y)=\{\psi_j^*(y)\}^t\cdot\{f(x,y_j^*)\} \qquad (4\cdot1\cdot21)$$

and $\{x_i^*,\ i=1,2,..,m^*\}$ are a finer division of the coarse range of x than $\{x_i,i=1,2,..,m\}$ and the points $\{y_j^*,j=1,2,..,n^*\}$ are a finer division of the coarse range of y than $\{y_j,j=1,2,..,n\}$. In all cases we assume that cardinality conditions hold:

$$\phi_i(x_s^*)=\delta_{is} \text{ and } \psi_j(y_t^*)=\delta_{jt}\begin{cases}s=1,2,..,m^*\\t=1,2,..,n^*\end{cases} \qquad (4\cdot1\cdot22)$$

Thus to compute the blending approximation we need the tensor products of vectors whose elements are basic functions (Lagrange polynomials or cardinal splines) and formation of inner products.

Further refinement of the mesh points corresponding to the two variables would lead to higher order blending[10] where the Boolean Sum approximation is built up by appropriately combining the various tensor product approximations[7]. Let $\{x_i^{**}, i=1,2,..,m^{**}\}$ be a division of the range of x finer than $\{x_i^*, i=1,2,..,m^*\}$ and $\{y_j^{**}, j=1,2,..,n^{**}\}$ be a still finer division of the range of y than $\{y_j^*, j=1,2,..,n^*\}$.

Then we compute the Boolean Sum $\Omega^{**}(x,y)$ as

$$[M_y^{**}L_x+M_yL_x^{**}+M_y^*L_x^*-M_y^*L_x-M_yL_x^*]f(x,y) \quad (4\cdot1\cdot23)$$

where $L_x^{**} f(x,y)=\{\phi_i^{**}(x)\}^t\cdot\{f(x_i^{**},y)\}$ (4·1·24)

$$M_y^{**} f(x,y)=\{\psi_j^{**}(y)\}^t\cdot\{f(x,y_j^{**})\} \quad (4\cdot1\cdot25)$$

with appropriate Cardinality conditions.

4·2 Cardinal Spline basis functions

In this paper, we will confine ourselves to approximations using Cardinal Splines-Splines satisfying certain orthogonality conditions, as they form excellent basis functions. Although there are several other equivalent forms of such splines, we intend using the powers and plusses form[6] as these are most suitable from the non-numerical data structure point of view; ofcourse the present data structure permits the absolutes form also. (Here we restrict ourselves to the problem of equispaced data points with a spacing h; using the transformation $y=(x-x_0)/h$, we can always bring the discussions to that of positive-integer-valued mesh points.)

Let m and n be integers such that $m+1 \geqslant n \geqslant 1$; put $q=2n-1$. For any function f(x) whose domain includes the integers 0,1,..,m there is a spline approximation $\Omega(x)$ based on the values $x_0, x_1,...,x_m$ and of type n:

$$\Omega(x) = \sum_{i=1}^{m} f(x_i)\cdot\beta^{m,i}(x) \quad (4\cdot2\cdot1)$$

where $\beta^{m,i}$ is the ith cardinal spline for m and n of the form,

$$\beta^{m,i}(x)= \sum_{\lambda=0}^{n-1} b_i^{m,i}\, x^{\lambda} + \sum_{\nu=0}^{m} C^{m,i}(x-\nu)_+^q$$

$$(i=0,1,2,..,m) \quad (4\cdot2\cdot2)$$

$$\text{where } t_+ = \begin{cases} t & \text{if } t \geqslant 0 \\ 0 & \text{if } t < 0 \end{cases} \quad (4\cdot2\cdot3)$$

The function $\beta^{m,i}(x)=\beta^{m,m-i}(m-x)(i=0,1,..m)$

$$\beta^{m,i}(\lambda)=\delta_{i\lambda} \begin{cases} =0 & \text{if } i \neq \lambda \\ =1 & \text{if } i = \lambda \end{cases} \quad (4\cdot2\cdot4)$$

Example: m = 2; n=2; q=2n-1 = 3.

$\beta^{2,0}(x)=1-1\cdot25x+\cdot25x_+^3-0\cdot5(x-1)_+^3+0\cdot25(x-2)_+^3$

$\beta^{2,1}(x)=1\cdot5x-0\cdot5x_+^3+(x-1)_+^3-0\cdot5(x-2)_+^3$

$\beta^{2,2}(x)=-0\cdot25x+0\cdot25x_+^3-0\cdot5(x-1)_+^3+0\cdot25(x-2)_+^3$

The case m=n-1 of $\beta^{m,i}(i=0,1,..,m)$(without the $(x-\nu)_+$ terms) correspond to Lagrangians. Here the spline approximation is

$$\Omega(x) = \sum_{i=0}^{m} f(x_i)L_i(x) \quad (4\cdot2\cdot5)$$

where

$$L_i(x)=\frac{(x-x_0)(x-x_1)..(\widehat{x-x_i})..(x-x_m)}{(x_i-x_0)(x_i-x_1)..(\widehat{x_i-x_i})..(x_i-x_m)} \quad (4\cdot2\cdot6)$$

where $\frown$ indicates the omission of the factor beneath it.

$L_i(x)$ assumes a form

$$L_i(x)=b_{i,0}+b_{i,1}x+..\ b_{i,m}\,x^m \quad (4\cdot2\cdot7)$$

without involving plusses or absolutes.

Example: m=2; n=3;

$L_0(x)=1-1\cdot5x+0\cdot5x^2=\beta^{2,0}(x)$

$L_1(x)=2x-x^2=\beta^{2,1}(x)$

$L_2(x)=-0\cdot5x+0\cdot5x^2=\beta^{2,2}(x)$

4·3 Example

The data points are given in Table 1. For this example, we have taken the coarse mesh points as x_i=0,1,2 and y_j=0,1,2 and fine mesh points are x_i = 0,0·5, 1, 1·5,2 and y_j=0, 0·5, 1, 1·5, 2 .

x	y	f(x,y)
0·0	0·0	3·000
0·0	0·5	5·500
0·0	1·0	8·000
0·0	1·5	10·500
0·0	2·0	13·000
0·5	0·0	4·250
0·5	0·5	6·875
0·5	1·0	9·500
0·5	1·5	12·125
0·5	2·0	14·750
1·0	0·0	6·000
1·0	0·5	9·000
1·0	1·0	12·000
1·0	1·5	15·000
1·0	2·0	18·000
1·5	0·0	8·250
1·5	0·5	11·875
1·5	1·0	15·500
1·5	1·5	19·125
1·5	2·0	22·750
2·0	0·0	11·000
2·0	0·5	15·500
2·0	1·0	20·000
2·0	1·5	24·500
2·0	2·0	29·000

Table 1

a. Lagrangian Basis Functions for $x = 0, 0\cdot 5, 1, 1\cdot 5, 2$ (Fine mesh)

$$\phi_1^*(x)=1-4\cdot 16x+5\cdot 83x^2-3\cdot 33x^3+0\cdot 66x^4$$
$$\phi_2^*(x)=8x-4\cdot 33x^2+12x^3-2\cdot 66x^4$$
$$\phi_3^*(x)=-6x+19x^2-16x^3+4x^4$$
$$\phi_4^*(x)=2\cdot 66x-9\cdot 33x^2+9\cdot 33x^3-2\cdot 66x^4$$
$$\phi_5^*(x)=-0\cdot 5x+1\cdot 83x^2-2x^3+0\cdot 66x^4$$

b. Lagrangian Basis Functions for $y=0,0\cdot 5, 1,1\cdot 5,2$ (Fine mesh)

$$\psi_1^*(y)=1-4\cdot 16y+5\cdot 83y^2-3\cdot 33y^3+0\cdot 66y^4$$
$$\psi_2^*(y)=8y-4\cdot 33y^2+12y^3-2\cdot 66y^4$$
$$\psi_3^*(y)=-6y+19y^2-16y^3+4y^4$$
$$\psi_4^*(y)=2\cdot 66y-9\cdot 33y^2+9\cdot 33y^3-2\cdot 66y^4$$
$$\psi_5^*(y)=-0\cdot 5y+1\cdot 83y^2-2y^3+0\cdot 66y^4$$

c. Cardinal spline basis functions for the points $x_i,y_j=0,1,2$ (Coarse mesh) [6]

$$\phi_1(x)=1-1\cdot 25x+\cdot 25(x)_+^3-\cdot 5(x-1)_+^3+\cdot 25(x-2)_+^3$$
$$\phi_2(x)=1\cdot 5x-\cdot 5(x)_+^3+(x-1)_+^3-\cdot 5(x-2)_+^3$$
$$\phi_3(x)=-\cdot 25x+\cdot 25(x)_+^3-\cdot 5(x-1)_+^3+\cdot 25(x-2)_+^3$$
$$\psi_1(y)=1-1\cdot 25y+\cdot 25(y)_+^3-\cdot 5(y-1)_+^3+\cdot 25(y-2)_+^3$$
$$\psi_2(y)=1\cdot 5y-\cdot 5(y)_+^3+(y-1)_+^3-\cdot 5(y-2)_+^3$$
$$\psi_3(y)=-\cdot 25y+\cdot 25(y)_+^3-\cdot 5(y-1)_+^3+\cdot 25(y-2)_+^3$$

d. Computation of $\Omega^*(x,y)$

$$\Omega^*(x,y)=(M_y^*L_x+M_yL_x^*-M_yL_x)f(x,y)$$

therefore takes the form

$$\Omega^*(x,y)=\sum_{i=1}^{3}\sum_{j=1}^{5}\phi_i(x)\psi_j^*(y)f(x_i,y_j^*)$$
$$+\sum_{i=1}^{5}\sum_{j=1}^{3}\phi_i^*(x)\psi_j(y)f(x_i^*,y_j)$$
$$-\sum_{i=1}^{3}\sum_{j=1}^{3}\phi_i(x)\psi_j(y)f(x_i,y_j)$$

$$=(\underset{1x3}{\phi^t(x)}\otimes\underset{1x5}{\psi^{t*}(y)}+\underset{1x5}{\phi^{*t}(x)}\otimes\underset{1x3}{\psi^t(y)}$$
$$-\underset{1x3}{\phi^t(x)}\otimes\underset{1x3}{\psi^t(y)})\,\underset{(15x1)}{\mathcal{V}(f(x_i^*,y_j^*))}\qquad(4\cdot 2\cdot 8)$$

Note: For the sake of conformability in multiplication, we introduce appropriate number of zeros at points corresponding to $x_i=0\cdot 5,1\cdot 5$ (or $y_j=0\cdot 5,1\cdot 5$), so that the third term in $(4\cdot 2\cdot 8)$ can be written as:

$$(\underset{1x5}{\phi'^t(x)}\otimes\underset{1x3}{\psi^t(y)})\text{ (or as }\underset{1x3}{\phi^t(x)}\otimes\underset{1x5}{\psi'^t(y)}).$$

$$\sum_{j=1}^{5}\psi_j^*(y)f(x_i,y_j^*)=5y+3,6y+6,\quad 9y+11$$
for $i=1,2,3$.

$$\sum_{j=1}^{3}\psi_j(y)f(x_i^*,y_j)=5y+3,5\cdot 25y+4\cdot 25,6y+6,$$
$7\cdot 25y+8\cdot 25,9y+11$, for $i = 1,2,3,4,5$.

$$\sum_{j=1}^{3}\psi_j(y)f(x_i,y_j)=5y+3,6y+6,\quad 9y+11$$
for $i=1,2,3$.

Therefore $\Omega^*(x,y)=3+2x+ x^2+ x^2y+5y$.

Remark: Note that

$\Omega(x,y)=M_yL_xf(x,y)$=Tensor product approximation

$$=3+2\cdot 5x+\cdot 5(x)_+^3-(x-1)_+^3+$$
$$+5y+\cdot 5xy+\cdot 5(x)_+^3y-(x-1)_+^3\,y$$

is not accurate enough.

5. Application to System Theory

In many applications in System Theory[11], it is necessary to compute the g.c.d. of two multivariable polynomials and also the inverse of matrices whose elements are m.v.p. The algorithms for g.c.d. are available in Knuth[12] Collins[1,2]; these can be implemented using the above data structure. The algorithm for inversion of matrices whose elements are polynomials is available in Krishnamurthy[13]. This algorithm can also be mechanized with the above data structure.

One particular difficulty that will be encountered in the above applications is the coefficient and degree growth of polynomials. It is possible to obviate this difficulty using the finite field transform techniques in which the integral coefficients of the m.v.p. are mapped to the elements of a finite field; the computations are then performed in this finite field with the same data structure and the result is mapped back to the set of integers. Unfortunately, due to lack of space, we are not able to describe these here in detail.

6. Concluding Remarks

It is interesting to note that the multivariable approximation methods can be realized using inner products and tensor products of vectors whose components are the basis functions such as Cardinal Splines. It is also easily seen that the tensor product can be obtained by a minor change in the algorithm used for computing the inner-product; accordingly a tensor product can be computed in n-inner product times (for n-component vectors) sequentially or in one-inner product time using parallelism.

Recently, there has been a trend for the design of inner product computers[14]; if the data structure described here is incorporated for such machines, the multivariable approximation can be computed economically using the blending function methods.

Since m.v.p. processing is required for several applications it seems worthwhile realizing all these algorithms in hardware.

References

1. G.E. Collins, 'PM, a system for polynomial manipulation', Comm. A.C.M., Vol. 9, No. 8, pp. 578-589, Aug. 1966.

2. G.E. Collins, 'Computer algebra of polynomials and rational functions', Amer. Math. Monthly, Vol. 180, pp. 725-755, 1973.

3. W.S. Brown, 'The ALPAK System for non-numerical algebra on a digital computer -I: Polynomials in several variables and truncated power series with polynomial coefficients', Bell Sys. Tech. J., Vol. 42, pp. 2081-2119, Sept.1963.

4. W.S. Brown, J.P. Hyde and B.A. Tague, 'The ALPAK System for non-numerical algebra on a digital computer-II: Rational functions of several variables and truncated power series with rational function coefficients', Bell Sys. Tech. J., Vol. 43, pp. 785-804, Mar. 1964.

5. J.P. Hyde, 'The ALPAK System for non-numerical algebra on a digital computer-III: Systems of linear equations and a class of side relations', Bell Sys. Tech. J., Vol. 43, pp. 1547-1562, July 1964.

6. A. Sard and S. Weintraub, 'A Book of Splines', John Wiley, N.Y. 1970.

7. W.J. Gordon, 'Distributive Lattices and the Approximation of Multivariate Functions' in Approximations with special emphasis on spline functions, edited by I.J. Schoenberg, Academic Press, N.Y. 1969.

8. M.J.D. Powell, 'Numerical methods for fitting functions of two variables', presented at the IMA Conference on the State-of-the-ART in Numerical Analysis, University of York, U.K., Apr. 1976.

9. L.L. Schumaker, 'Fitting surfaces to scattered data', Proc. Conf. on Approximation Theory, University of Texas, Eds. C.K. Chui, G.G. Lorentz and L.L. Schumaker, Academic Press, N.Y. 1977.

10. F.J. Delvos and H. Pasdorf, 'Nth order blending' in constructive theory of functions of several variables', Lecture notes in Mathematics 571, Springer Verlag, N.Y. 1977.

11. N.K. Bose, 'A criterion to determine if two multivariable polynomials are relatively prime', Proc. IEEE, Vol. 60, pp. 134-135, 1972.

12. D.E. Knuth, 'The Art of Computer Programming', Vol. 2: Semi-numerical algorithms, Addison-Wesley Pub., 1969.

13. E.V. Krishnamurthy, 'Exact inversion of a rational polynomial matrix using finite field transforms', SIAM J. of App. Maths., Vol. 35, No. 3, Nov.1978 (to appear).

14. E.E. Swartzlander, B.K. Gilbert and I.S. Reed, 'Inner Product computers', IEEE Trans. Computers, Vol. C-27, No. 1, pp. 21-31, Jan. 1978.

Acknowledgements

A part of this work was carried out during the first author's visit to the Computer Science Center, University of Maryland, during May - June 1978. The support of the National Science Foundation, U.S.A. under grant MCS-76-23763 is gratefully acknowledged.