Novi Sad J. Math.
Vol. 28, No. 3, 1998, 137-143

DIVIDING THE POLYNOMIAL EXPRESSIONS BASED ON THE CRITERIA OF REDUCING THE NUMBER OF COMPUTATIONAL OPERATIONS

Miloš Racković[1]
University of Novi Sad, Faculty of Science, Institute of Mathematics
Trg D. Obradovića 4, 21000 Novi Sad, Yugoslavia

Abstract

Dividing the appropriate polynomial expressions into the products is a basic part of the algorithm for reducing the number of computational operations in the analytical expressions of the robotic mechanisms dynamic models. On the basis of the mathematical background, the data structures and algorithm for dividing these polynomial expressions are developed. The central topic of the paper is the algorithm for choosing the products which give the largest reduction in the number of computational operations. From all generated products, that one is chosen which gives "maximal" reduction in the number of operations. Then, from all other products, the chosen addends are eliminated and the same procedure is repeated. The procedure stops when there is no product left to be chosen.

AMS Mathematics Subject Classification (1991): 68Q40, 70B15
Key words and phrases: polynomial expressions, reduction in the number of operations.

[1]The work is supported by Science Fund of Serbia, through the Institute of Mathematics, Novi Sad

1. Introduction

Mathematical modelling of the complex systems generates the analytical expressions with great calculating complexity which should be reduced in order to obtain the efficient model. The process of reducing calculating complexity of the analytical expressions is one of the basic processes in the generation of complete mathematical model. In [1], the complete procedure for generating the symbolic models of the dynamics of complex robotic mechanisms is described, and in [2] is given the structural system analysis of the process of reducing the calculating complexity.

All the analytical expressions obtained in [1] are of the polynomial form

$$Y = \sum_{i=1}^{I} k_i \cdot \prod_{l=1}^{L} x_l^{e_{il}} \tag{1}$$

where:

Y - is the variable to be calculated;

k_i - is a constant coefficient related to the i-th addend;

x_l - is one of the basic variables of the robotic system model represented by its name ($q, \dot{q}, \ddot{q}, \sin q, \cos q$, where q represents the degree of freedom). For each addend the same sequence of variables x_j, $j = 1, ..., L$ is used.

e_{il} - is the exponent of the l-th variable of the i-th addend. The algorithm for forming the mathematical model ensures that each of the exponents is a non-negative integer number.

The main task is to form the calculation graph for the chosen analytical expressions of the type (1), with the least number of mathematical operations. To obtain a maximal reduction in the number of mathematical operations it has been proposed in [1-5] to divide the expressions in the following way

$$Y = \sum_{w=1}^{W} (Y_{w1} \cdot Y_{w2}) + Y_{W+1} \tag{2}$$

where Y_{w1}, Y_{w2}, $w = 1, ..., W$ and Y_{W+1} are also the expressions of the type (1).

The expressions Y_{l1}, Y_{l2}, $l = 1, ..., W$ have two addends at least, and are determined in a way which maximises reduction of the number of mathematical operations. Y_{W+1} represents the remainder of the expression Y which

can not be divided into products any more. After dividing the expressions into the products the monomial extraction algorithm is applied to form the calculating graph for the chosen expressions [1].

This paper gives the mathematical basis for dividing the expressions into the products. The paper also describes the data structures and the algorithms for choosing these products which give the "maximal" reduction in the number of computational operations.

2. Background

The problem of dividing an expression into the products is analogous to the problem of finding the structural matrices [6] A and B which satisfy the equation

$$A \cdot B = C \tag{3}$$

for the given structural matrix C, and can be represented by the following system of equations

$$e_i^A + e_j^B = e_m^C \; ; \; k_i^A \cdot k_j^B = k_m^C \tag{4}$$

$$m = (i-1) \cdot J + j, \quad i = 1, ..., I; j = 1, ..., J.$$

In [1, 3], it has been shown that satisfying equations (5) for all different $m1$, $m2$, $m3$, $m4 \in \{1, ..., M\}$ gives the necessary and sufficient condition for the existence of the solution of the system (4).

$$e_{m1}^C - e_{m3}^C = e_{m2}^C - e_{m4}^C \; ; \; \frac{k_{m1}^C}{k_{m3}^C} = \frac{k_{m2}^C}{k_{m4}^C} \tag{5}$$

This claim was proved by deriving a solution which satisfies the system (4).

This mathematical analysis serves as a basis for constructing the algorithm for generating the products, the candidates for dividing the expressions [7]. The first step is to represent the expression by the structural matrix. The next step is generating the equations of the form (5) for different combinations of the addends. On the basis of the starting products, we then construct all possible products of greater dimensions. By this procedure all the products, candidates for dividing, are generated.

Before dividing the expressions, those products which give a "maximal" reduction in the number of computational operations should be chosen. The quotation marks are used because this choice is made using a suboptimal algorithm. ¿From all generated products that one is chosen which gives "maximal" reduction in the number of operations. Then, from all other products, the chosen addends are eliminated, and the procedure is repeated again. The procedure stops when there is no product left to be chosen. It is obvious that this greedy algorithm, which in every step takes the "best" solution, does not guarantee the globally best solution. Still, this algorithm can be effectively implemented and gives satisfactory results in practical applications.

3. Choice of the products

All products, the candidates for dividing the expressions, were generated by the algorithm described in [7]. These products are stored in the list of products *lispro*. This is a list of structures which represents the products. The data structure which describes one product is given in the pseudocode.

```
struct product {
 int m, n, matadd[m][n];
 struct product * nextpr;
}
```

Dimension of the product is represented by m and n and equals $m \times n$. The matrix $matadd[m][n]$ represents the indexes of the addends which form the given product. In *nextpr*, the pointer to the next product in the list of products is given. An array of pointers *arrpp* with the dimension *dap* is introduced, where each member of this array points to the first element in the list of products which has the corresponding value for m. ¿From the list of products *lispro* those products are chosen which give a "maximal" reduction in the number of operations. The procedure *ChoBestPro* is given in the pseudocode.

```
ChoBestPro(lispro, arrpp, dap) {
 p = lispro;
```

```
 while (p! =NULL) {
  while (p! = arrpp[dap − 1])
   p = p− > nextpr;
  MaxDim(p);
  p3 = ReconPro(lispro, p, arrpp, dap);
  InsLisPro(nlispro, p3);
  DelProLis(p3, lispro, narrpp, dap);
  p = lispro;
 }
 lispro = nlispro;
}
```

The first step of this procedure is positioning of the variable p onto the first product with the maximal value for m. This is the product pointed by the pointer $arrpp[dap - 1]$. Then, among the products with the same value for m, the variable p is positioned by the procedure $MaxDim(p)$, onto that product having maximal dimension $m \times n$. This product is eliminated from the list of the products *lispro* by the procedure *ReconPro*(*lispro*, *p*, *arrpp*, *dap*), which reconnects the list, and if necessary, modifies the array of pointers *arrpp* and its dimension *dap*. Then, the chosen product $p3$ (one with the maximal dimension) is inserted into the new list of products *nlispro* by the procedure $InsLisPro(nlispro, p3)$. In this new list are stored those products which are chosen for dividing the expression.

The next step is to modify all the products that remained in the list *lispro*, which is achieved by applying the procedure *DelProLis*(*p3*, *lispro*, *arrpp*, *dap*). By this procedure, those addends which participate in the chosen product $p3$ are eliminated from all other products, because in the process of dividing the expression every expression addend can be used only once. In this way, some of the products are completely eliminated from the list of products and some of them obtain a new, smaller dimension. In the case of eliminating a product from the list, the array of pointers *arrpp* and its dimension *dap* are modified if necessary. In the case of changing the dimension, the value for m can be smaller than before, and the product must be transferred to the corresponding place in the list *lispro*. Then, the variable p is positioned again onto the beginning of the list *lispro*, and the same procedure is repeated until the list of products *lispro* is empty. Finally, the pointer is set onto the new list *nlispro* in which all the chosen products are stored.

It is obvious that the above algorithm does not guarantee the optimality of the chosen products in the sense of the reduction in the number of computational operations. Instead of the optimal one, some kind of a greedy algorithm is applied here. In each step, the algorithm chooses that product which gives a "maximal" reduction in the number of operations. Again, the algorithm does not calculate the real reduction in the number of operations for each of the products, with the aim of obtaining the one which gives a maximal reduction. In order to construct the procedure for calculating the real reduction for each product, it is necessary to keep information about the number of factors in every addend in addition to that about the number of addends in the product. Such solution demands more complex data structure for the list of products (*lispro*) and additional time for calculating the reduction in the number of operations. Instead of that, for the product with "maximal" reduction, that one is chosen which has a maximal dimension $m \times n$ (number of the addends in the product) among the products having the same maximal value for m. We should mention that the products are arranged so that m is always less or equal than n. The idea of the algorithm is now clear. It is better to choose the product with a smaller dimension but with m and n closer to each other (4×5), than the one with a greater dimension but with a larger distance between m and n (2×11).

This algorithm sacrifices the correctness in the calculating the reduction in the number of operations in order to obtain a fast and still very efficient procedure for choosing the products for dividing the polynomial expressions.

4. Conclusion

The algorithms for forming symbolic mathematical models of complex robotic systems have been developed. With the aim of reducing the number of mathematical operations, the investigations are directed towards the development of the algorithms for simplification of analytical expressions. These algorithms form the process for reduction of calculating complexity of the analytical expressions which is a necessary part of the mathematical modelling process in symbolic form. The aim of reducing the model calculating complexity is the need for calculating numerical values of the model quantities in the real-time regime.

This paper gives a new approach to the reduction of the model calculating complexity, which instead of using the specific features of robotic

mechanisms and the corresponding equations for deriving their models, determines the mathematical form of the model analytical expressions and performs the general procedures for reducing their calculating complexity. This approach is illustrated by developing an algorithm for choosing the products with a "maximal" reduction in the number of computational operations. This choice is made using a suboptimal algorithm, which in an efficient way determines which products should be chosen.

References

[1] Racković, M., *Generation of the Mathematical Models of Complex Robotic Mechanisms in Symbolic Form*, Ph. D. Thesis, University of Novi Sad, 1996.

[2] Racković, M., Vukobratović, M., Surla, D., *Generation of Dynamic Models of Complex Robotic Mechanisms in Symbolic Form*, Robotica (in press), 1997.

[3] Racković, M., Surla, D., *The Algorithms and Data Structures for Forming Symbolic Models of the Robotic Systems*, FILOMAT (Niš) **9:3** (1995), pp. 887–897.

[4] Surla, D., Racković, M., *Forming a Calculation Graph for the Polynomial Expressions*, X Conference on Applied Mathematics PRIM'95, Budva (1995), pp. 275–282.

[5] Racković, M., Surla, D., *Reducing the Calculating Complexity of the Robotic Mechanism Dynamic Model*, Third ECPD International Conference on Advanced Robotics, Intelligent Automation and Active Systems, September 15-17, 1997, Bremen, Germany, pp. 107-112.

[6] Vukobratović, M., Kirćanski, N., *Real-Time Dynamics of Manipulation Robots*, Springer-Verlag, 1985.

[7] Racković, M., *Generating the Products, Candidates for Dividing the Polynomial Expressions*, Proceedings of the VIII Conference on Logic and Computer Science, Novi Sad, Yugoslavia, September 1-4, 1997. pp. 183-189.