ISSN 1342-2804

Research Reports on
Mathematical and
Computing Sciences

Efficient Evaluation of Polynomials and Their Partial Derivatives in Homotopy Continuation Methods

Masakazu Kojima

August 2006, B–433

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES B: Operations Research

B-433 Efficient Evaluation of Polynomials and Their Partial Derivatives in Homotopy Continuation Methods

Masakazu Kojima†, August 2006

Abstract.
The aim of this paper is to study how efficiently we evaluate a system of multivariate polynomials and their partial derivatives in homotopy continuation methods. Our major tool is an extension of the Hornor scheme, which is popular in evaluating a univariate polynomial, to a multivariate polynomial. But the extension is not unique, and there are many Hornor factorizations of a given multivariate polynomial which require different numbers of multiplications. We present exact method for computing a minimum Hornor factorization, which can process smaller size polynomials, as well as heuristic methods for a smaller number of multiplicatios, which can process larger size polynomials. Based on these Hornor factorization methods, we then present methods to evaluate a system of multivariate polynomials and their partial derivatives. Numerical results are shown to verify the effectiveness and the efficiency of the proposed methods.

Key words.
Hornor scheme, multivariate polynomial, homotopy continuation method

† Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan. *kojima@is.titech.ac.jp*

1 Introduction

Various types of homotopy continuation methods, such as the linear homotopy continuation method [1, 3, 4, 8, 10] and the polyhedral homotopy continulation method [2, 7, 9, 15, 18, 19], have been studied extensively as numerical methods for computing all isolated solutions of a system of polynomial equations in multi complex variables. When we trace homotopy curves using a predictor-corrector procedure, we need to evaluate polynomials and their derivatives repeatedly along the homotopy curves. Thus, how fast we evaluate polynomials and their derivatives is a key to efficient implementation of homotopy continuation methods. There are lots of software packages [5, 6, 16, 20, 21] for homotopy continuation methods. Some techniques for efficient evaluation of polynomials and their derivatives must have been used there. To the best of the author's knowledge[1], however, there have been no general and/or rigorous discussion on efficient evaluation of polynomials and their derivatives. The main purpose of this paper is to study the subject from an optimization point of view, *i.e*, how we minimize the number of multiplications in evaluation of polynomials and their derivatives.

To describe the subject of the paper more precisely, let us introduce some symbols and notation. Let $\mathbb{C}$ and $\mathbb{Z}_+$ denote the set of complex numbers and the set of nonnegative integers, respectively. For every vector variable $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{C}^n$ and every $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{Z}^n_+$, we use the notation $\boldsymbol{x}^{\boldsymbol{\alpha}}$ for the monomial $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$. Then we can write a polynomial φ in $\boldsymbol{x} \in \mathbb{C}^n$ as $\varphi(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} c(\boldsymbol{\alpha})\ \boldsymbol{x}^{\boldsymbol{\alpha}}$ for some finite subset $\mathcal{A}$ of $\mathbb{Z}^n_+$ and some $c(\boldsymbol{\alpha}) \in \mathbb{C}$ ($\boldsymbol{\alpha} \in \mathcal{A}$). We consider a general coefficient-parameter homotopy function $\boldsymbol{h} : \mathbb{C}^n \times [0, 1] \to \mathbb{C}^n$ [11], which covers linear and polyhedral homotopy functions as special cases, such that

$$
\begin{aligned}
\boldsymbol{h}(\boldsymbol{x}, t) &= (h_1(\boldsymbol{x}, t), h_2(\boldsymbol{x}, t), \ldots, h_n(\boldsymbol{x}, t)), \\
h_j(\boldsymbol{x}, t) &= \sum_{\boldsymbol{\alpha}\in\mathcal{A}_j} c_{j,\boldsymbol{\alpha}}(t)\boldsymbol{x}^{\boldsymbol{\alpha}}\ (j = 1, 2, \ldots, n).
\end{aligned}
\tag{1}
$$

Here each $c_{j,\boldsymbol{\alpha}}(t)$ is continuously differentiable with respect to $t \in [0, 1]$. When we numerically trace solution curves of $\boldsymbol{h}(\boldsymbol{x}, t) = \boldsymbol{0}$ by the predictor-corrector procedure, we evaluate values of $h_j(\boldsymbol{x}, t)$, $\partial h_j(\boldsymbol{x}, t)/\partial t$ and $\partial h_j(\boldsymbol{x}, t)/\partial x_i$ ($i = 1, 2, \ldots, n, j = 1, 2, \ldots, n$) at each iteration. These are all polynomials in $\boldsymbol{x} \in \mathbb{C}^n$. Tracing one solution curve often requires more than hundreds of evaluations of them, so that their fast evaluation over all solution curves is crucial to an efficient implementation of a homotopy continuation method.

We focus our attention to the number of multiplications required to evaluate $h_j(\boldsymbol{x}, t)$, $\partial h_j(\boldsymbol{x}, t)/\partial t$ and $\partial h_j(\boldsymbol{x}, t)/\partial x_i$ ($i = 1, 2, \ldots, n, j = 1, 2, \ldots, n$). We assume that all coefficients $c_{j,\boldsymbol{\alpha}}(t)$ and their derivatives $dc_{j,\boldsymbol{\alpha}}(t)/dt$ ($\boldsymbol{\alpha} \in \mathcal{A}_j, j = 1, 2, \ldots, n$) are generic nonzero complex (or real) numbers when $h_j(\boldsymbol{x}, t)$, $\partial h_j(\boldsymbol{x}, t)/\partial t$ and $\partial h_j(\boldsymbol{x}, t)/\partial x_i$ ($i = 1, 2, \ldots, n, j = 1, 2, \ldots, n$) are evaluated. Thus the number of multiplications required depends only on the support set $\mathcal{A}_j$ but not on specific values of the coefficients and their derivatives. Therefore, this leads to a simpler question how we minimize the number of multiplications in evaluating

[1] In numerical experiments, HOM4PS [5] worked on some benchmark polynomials much faster than PHoM [6]. Probably, this difference is mainly due to a difference in evaluation of polynomials and their derivatives.

a single multivariate polynomial

$$f(\boldsymbol{x}) \;=\; \sum_{p=1}^{m} c_p \boldsymbol{x}^{\boldsymbol{\alpha}_p}, \tag{2}$$

where m denotes a positive integer, $\boldsymbol{\alpha}_p \in \mathbb{Z}^n_+$ and c_p a generic nonzero complex (or real) number. Note that each c_p is corresponding to $c_{j,\boldsymbol{\alpha}}(t)$ when we evaluate $h_j(\boldsymbol{x}, t)$ and to $\partial c_{j,\boldsymbol{\alpha}}(t)/\partial t$ when we evaluate $dh_j(\boldsymbol{x}, t)/dt$.

One of our major tools is a multivariate Hornor scheme. The Honor scheme is a popular and standard technique which has been frequently used for evaluating a polynomial in a single variable. The scheme is known to be very effective to reduce the roundoff error which would occur if the monomials were evaluated separately and added up [22]. The idea of the Hornor scheme is naturally extended to multivariate polynomials, but its extension is not unique; in general, there are many distinct "Hornor factorizations" of a given multivariate polynomial, which require different numbers of multiplicaions. As an example, let us consider a polynomial in $\boldsymbol{x} = (x_1, x_2) \in \mathbb{C}^2$:

$$f(\boldsymbol{x}) \;=\; c_1x_1^3 + c_2x_1^5x_2^3 + c_3x_1^4x_2^4 + c_4x_2^2 + c_5. \tag{3}$$

In this case, some different Hornor factorizations are:

$$\begin{aligned} f(\boldsymbol{x}) &= x_1^3(c_1 + c_2x_1^2x_2^3) + x_2^2(c_3x_1^4x_2^2 + c_4) + c_5, \\ f(\boldsymbol{x}) &= c_1x_1^3 + x_2^2(x_1^4x_2(c_2x_1 + c_3x_2) + c_4) + c_5, \\ f(\boldsymbol{x}) &= x_1^3(c_1 + x_1x_2^3(c_2x_1 + c_3x_2)) + c_4x_2^2 + c_5, \end{aligned} \tag{4}$$

which require 16, 12 and 11 multiplications to evaluate $f(\boldsymbol{x})$, respectively. Thus, the problem of finding *a minimal Hornor factorization* (a Honor factorization with the minimum number of multiplications) arises. As in the single variable case, the use of the multivariate Hornor scheme results in less roundoff errors. This was discussed in details in the papers [12, 13]. The current paper focusses its attention to the number of multiplications in factorizations of a polynomial generated by the multivariate Hornor scheme, and discusses the problem of minimizing the number of multiplications over all Honor factorizations.

A method for a Hornor factorization of a single polynomial with the minimum number (or a smaller number) of multiplications, however, is not enough to minimize the total number of multiplications in evaluating values of $h_j(\boldsymbol{x}, t)$, $\partial h_j(\boldsymbol{x}, t)/\partial t$ and $\partial h_j(\boldsymbol{x}, t)/\partial x_i$ ($i = 1, 2, \ldots, n, j = 1, 2, \ldots, n$) in (1) which are done at each iteration of the predictor-corrector procedure of a homotopy continuation method. Suppose that a component $h_j(\boldsymbol{x}, t)$ is factorized as in the right hand side of (4), where we may assume that c_i ($i = 1, 2, \ldots, 5$) are continuously differentiable functions in t. To evaluate $h_j(\boldsymbol{x}, t)$ using its Hornor factorization (4), we need to compute the monomials x_1^3, $x_1x_2^3$ and x_2^2. When we counted the number of multiplications required by the Hornor factorization (4) above, we assumed that these monomials are computed independently. But we can utilize the value of the third monomial x_2^2 to compute the second monomial $x_1x_2^3$ such that $x_1x_2^3 = x_1x_2 \times x_2^2$; hence we can save a multiplication. In general, many different monomials are generated in Hornor factorizations of the homotopy polynomials $h_j(\boldsymbol{x}, t)$ ($j = 1, 2, \ldots, n$) and also in evaluation of their partial derivatives $h_j(\boldsymbol{x}, t)$, $\partial h_j(\boldsymbol{x}, t)/\partial t$ and $\partial h_j(\boldsymbol{x}, t)/\partial x_i$ ($i = 1, 2, \ldots, n, j = 1, 2, \ldots, n$) by using the Hornor factorizations. We could consider the problem of minimizing the total number of multiplications to evaluate the homotopy functions and their partial derivatives simultaneously taking account of evaluation of those monomials. But this problem is too complicated and too difficult to solve exactly, so that we divide the problem into two phases. In the first phase, we find a minimum Hornor factorization or a Hornor factorization with a small number of multiplications for each component homotopy function separately, and then, in the second phase, we deal with the problem of minimizing the number multiplications to evaluate the set of monomials which are involved in the Hornor factorizations of the homotopy functions and in evaluation of their partial derivatives. The latter problem is also difficult to solve exactly, so that we propose a heuristic method.

The paper is organized as follows. In Section 2, we present a basic framework for the Hornor scheme for a polynomial after introducing symbols, notation and an illustrative example of a multivariate polynomial that we use through out the paper. Section 3 describes numerical methods for a minimum Hornor factorization of a single polynomial, and Section 4 three heuristic methods for Hornor factorizations of a single polynomial with a smaller number of multiplications. Based on the minimum and heuristic Hornor factorization methods given for a single polynomial in Sections 3 and 4, Section 5 discusses how efficiently we compute a system of polynomials and their partial derivatives. Section 6 is devoted to numerical results to compare the methods proposed in this paper.

2 A framework for the Hornor scheme for multivariate polynomials

2.1 Symbols and notation

We use the notation $\mathbb{C}[\boldsymbol{x}]$ for the ring of polynomials in $\boldsymbol{x} \in \mathbb{C}^n$. Associated with the polynomial $f \in \mathbb{C}[\boldsymbol{x}]$ given in (2), let

$$M = \{1, 2, \dots, m\} \ \text{ and } \mathcal{F}_p = \{\boldsymbol{\beta} \in \mathbb{Z}^n_+ : \boldsymbol{\beta} \leq \boldsymbol{\alpha}_p\} \ (p \in \boldsymbol{M}).$$

We introduce a family $\mathbb{F}[\boldsymbol{x}, f]$ of polynomials induced from the polynomial f of the form (2) such that

$$\mathbb{F}[\boldsymbol{x}, f] \ = \ \left\{ g \in \mathbb{C}[\boldsymbol{x}] : \begin{array}{c} \exists \text{nonempty } P \subseteq M \text{ and } \boldsymbol{\beta}_p \in \mathcal{F}_p \ (p \in P) \\ \text{such that } g(\boldsymbol{x}) = \displaystyle\sum_{p \in P} c_p \boldsymbol{x}^{\boldsymbol{\beta}_p} \end{array} \right\}.$$

Some members of $\mathbb{F}[\boldsymbol{x}, f]$ will serve as element polynomials of a Hornor factorization of the polynomial f. Let g be a polynomial in $\mathbb{F}[\boldsymbol{x}, f]$. Then there exists a nonempty $P \subseteq M$ and $\boldsymbol{\beta}_p \in \mathcal{F}_p$ $(p \in P)$ such that

$$g(\boldsymbol{x}) = \sum_{p \in P} c_p \boldsymbol{x}^{\boldsymbol{\beta}_p}. \tag{5}$$

Since the values of c_p $(p \in M)$ are not relevant througout the paper, the information $(P \subseteq M, \boldsymbol{\beta}_p \in \mathcal{F}_p \ (p \in P))$, which we will simply write as $(P, \boldsymbol{\beta}_p)$, is enough to describe a polynomial $g \in \mathbb{F}[\boldsymbol{x}, f]$. When $g \in \mathbb{F}[\boldsymbol{x}, f]$ is of the form (5), we identify $g \in \mathbb{F}[\boldsymbol{x}, f]$

as $(P, \boldsymbol{\beta}_p)$ and write $g = (P, \boldsymbol{\alpha}_p) \in \mathbb{F}[\boldsymbol{x}, f]$. In particular,we write the polynomial of the form (2) as $f = (M, \boldsymbol{\alpha}_p)$. We also identify a monomial $\boldsymbol{x}^{\boldsymbol{\beta}}$ with its power vector $\boldsymbol{\beta} \in \mathbb{Z}^n_+$. Let $\deg(\boldsymbol{x}^{\boldsymbol{\beta}}) = \deg(\boldsymbol{\beta}) = \sum_{i=1}^n [\boldsymbol{\beta}]_i$ denote the degree of a monomial, and $\text{t.deg}(g) = \text{t.deg}(P, \boldsymbol{\beta}_p) = \sum_{p \in P} \deg(\boldsymbol{\beta}_p)$ the total degree of a polynomial $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$.

2.2 Example 1

As an illustrative example, we consider a polynomial

$$f(\boldsymbol{x}) = c_1x_1x_2x_3x_4 + c_2x_2x_3x_4x_5 + c_3x_1x_3x_4x_5 + c_4x_1x_2x_4x_5 + c_5x_1x_2x_3x_5 + c_6. \quad (6)$$

In this case, we have

$$\begin{aligned}
M &= \{1, 2, 3, 4, 5, 6\}, \\
\boldsymbol{\alpha}_1 &= (1, 1, 1, 1, 0),\ \mathcal{F}_1 = \{\boldsymbol{\alpha} \in \{0, 1\}^5 : [\boldsymbol{\alpha}]_5 = 0\}, \\
\boldsymbol{\alpha}_2 &= (0, 1, 1, 1, 1),\ \mathcal{F}_2 = \{\boldsymbol{\alpha} \in \{0, 1\}^5 : [\boldsymbol{\alpha}]_1 = 0\}, \\
\boldsymbol{\alpha}_3 &= (1, 0, 1, 1, 1),\ \mathcal{F}_3 = \{\boldsymbol{\alpha} \in \{0, 1\}^5 : [\boldsymbol{\alpha}]_2 = 0\}, \\
\boldsymbol{\alpha}_4 &= (0, 1, 0, 1, 1),\ \mathcal{F}_4 = \{\boldsymbol{\alpha} \in \{0, 1\}^5 : [\boldsymbol{\alpha}]_3 = 0\}, \\
\boldsymbol{\alpha}_5 &= (1, 1, 1, 0, 1),\ \mathcal{F}_5 = \{\boldsymbol{\alpha} \in \{0, 1\}^5 : [\boldsymbol{\alpha}]_4 = 0\}, \\
\boldsymbol{\alpha}_6 &= (0, 0, 0, 0, 0),\ \mathcal{F}_6 = \{(0, 0, 0, 0, 0)\}.
\end{aligned}$$

If we take $P = \{1, 3, 4\}$, $\boldsymbol{\beta}_1 = (1, 1, 1, 0, 0) \in \mathcal{F}_1$, $\boldsymbol{\beta}_3 = (1, 0, 1, 0, 1) \in \mathcal{F}_3$ and $\boldsymbol{\beta}_4 = (0, 1, 0, 0, 1) \in \mathcal{F}_4$, then $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ turns out to be

$$g(\boldsymbol{x}) = c_1x_1x_2x_3 + c_3x_1x_3x_5 + c_4x_2x_5. \quad (7)$$

We see that

$$\begin{aligned}
\deg(\boldsymbol{x}^{\boldsymbol{\beta}_1}) &= \deg(x_1x_2x_3) = 3, \\
\deg(\boldsymbol{x}^{\boldsymbol{\beta}_3}) &= \deg(x_1x_3x_5) = 2, \\
\deg(\boldsymbol{x}^{\boldsymbol{\beta}_4}) &= \deg(x_2x_5) = 2, \\
\text{t.deg}(g) &= \text{t.deg}(P, \boldsymbol{\beta}_p) = \deg(\boldsymbol{x}^{\boldsymbol{\beta}_1}) + \deg(\boldsymbol{x}^{\boldsymbol{\beta}_2}) + \deg(\boldsymbol{x}^{\boldsymbol{\beta}_3}) = 7.
\end{aligned}$$

Applying the Hornor scheme which will be discussed later, we have some Hornor factorizations of f given in (6):

$$x_4(x_3(x_2(c_1x_1 + c_2x_5) + c_3x_1x_5) + c_4x_1x_2x_5) + c_5x_1x_2x_3x_5 + c_6, \quad (8)$$
$$x_4(x_2x_3(c_1x_1 + c_2x_5) + x_1x_5(c_3x_3 + c_4x_2)) + c_5x_1x_2x_3x_5 + c_6, \quad (9)$$
$$x_2x_3x_4(c_1x_1 + c_2x_5) + x_1x_5(x_4(c_3x_3 + c_4x_2) + c_5x_2x_3) + c_6, \quad (10)$$
$$x_3x_4(x_2(c_1x_1 + c_2x_5) + c_3x_1x_5) + x_1x_2x_5(c_4x_4 + c_5x_3) + c_6. \quad (11)$$

The Hornor factorizations (8) and (9) require 14 and 13 multiplications, respectively, while either of (10) and (11) requires 12 multiplications.

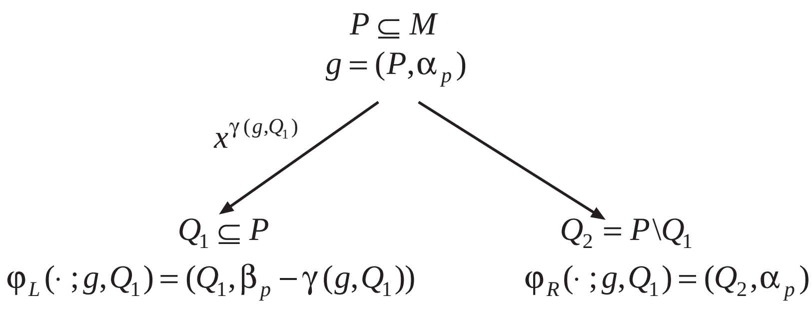

Figure 1: A partial-factorization of a polynomial $g \in \mathbb{F}[\boldsymbol{x}, f]$

2.3 Partial factorization

Let $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$. For every $Q \subseteq P$, we define

$$\begin{aligned} \gamma(g, Q)_i &= \min\{[\boldsymbol{\beta}_p]_i : \boldsymbol{\beta}_p \ (p \in Q)\} \in \mathbb{Z}_+ \ (i = 1, 2, \ldots, n), \\ \boldsymbol{\gamma}(g, Q) &= (\gamma(g, Q)_1, \gamma(g, Q)_2, \ldots, \gamma(g, Q)_n) \in \mathbb{Z}^n_+. \end{aligned}$$

Let

$$\mathcal{Q}(g) = \{Q \subseteq P : \#Q \geq 2, \ \boldsymbol{\gamma}(g, Q) \neq \mathbf{0}\}.$$

As an example, suppose that $g = (P, \boldsymbol{\beta}_p)$ is of the form (7). Then

$$\left.\begin{array}{l} P = \{1, 3, 4\}, \ \boldsymbol{\gamma}(g, \{1, 3, 4\}) = (0, 0, 0, 0, 0), \ \boldsymbol{\gamma}(g, \{1, 3\}) = (1, 0, 1, 0, 0), \\ \boldsymbol{\gamma}(g, \{1, 4\}) = (0, 1, 0, 0, 0), \ \boldsymbol{\gamma}(g, \{3, 4\}) = (0, 0, 0, 0, 1), \\ \mathcal{Q}(g) = \{\{1, 3\}, \ \{1, 4\}, \ \{3, 4\}\}. \end{array}\right\} \tag{12}$$

If $\mathcal{Q}(g) \neq \emptyset$, we say that $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ is *partially-factorizable*. In this case, if we take a $Q \in \mathcal{Q}(g)$, we have *a partial factorization of* g such that

$$\left.\begin{array}{l} g(\boldsymbol{x}) = \boldsymbol{x}^{\boldsymbol{\gamma}(g,\, Q)} \varphi_L(\boldsymbol{x}; g, Q) + \varphi_R(\boldsymbol{x}; g, Q), \\ \varphi_L(\cdot; g, Q) = (Q, \boldsymbol{\beta}_p - \boldsymbol{\gamma}(g, Q)) \text{ or} \\ \varphi_L(\boldsymbol{x}; g, Q) = \displaystyle\sum_{p \in Q} c_p \boldsymbol{x}^{\boldsymbol{\beta}_p - \boldsymbol{\gamma}(g,\, Q)}, \\ \varphi_R(\cdot; g, Q) = (P \backslash Q, \boldsymbol{\beta}_p) \text{ or} \\ \varphi_R(\boldsymbol{x}; g, Q) = \displaystyle\sum_{p \in P \backslash Q} c_p \boldsymbol{x}^{\boldsymbol{\beta}_p}. \end{array}\right\} \tag{13}$$

Here Q can coincide with P; in such a case, we assume that $\varphi_R(\cdot; g, Q) = 0$. Figure 1 illustrates a partila-factorization of a polynomial $g = (P, \boldsymbol{\beta}_p)$ in terms of a tree. If $\mathcal{Q}(g) = \emptyset$, we say that $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ is *non-factorizable*. Specifically, any Q with only one element is non-factorizable.

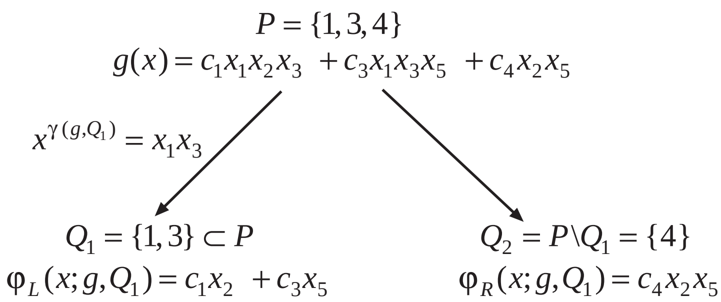

Figure 2: A partial-factorization of a polynomial $g \in \mathbb{F}[\boldsymbol{x}, f]$ given in (5)

In the case of (7), if we take $Q = \{1, 3\}$ then we have

$$\begin{aligned}
\boldsymbol{\gamma}(g, Q) &= (1, 0, 1, 0, 0),\ \boldsymbol{x}^{\boldsymbol{\gamma}(g, Q)} = x_1x_3, \\
\varphi_L(\boldsymbol{x}; g, Q) &= c_1x_2 + c_3x_5, \\
\varphi_R(\boldsymbol{x}; g, Q) &= c_4x_2x_5, \\
g(\boldsymbol{x}) &= \boldsymbol{x}^{\boldsymbol{\gamma}(g, Q)}\varphi_L(\boldsymbol{x}; g, Q) + \varphi_R(\boldsymbol{x}; g, Q) \\
&= x_1x_3(c_1x_2 + c_3x_5) + c_4x_2x_5.
\end{aligned}$$

See Fingure 2.

2.4 Multivariate Hornor factorizations

Note that both $\varphi_L(\cdot; g, Q)$ and $\varphi_R(\cdot; g, Q)$ in (13) belong to the class $\mathbb{F}[\boldsymbol{x}, f]$ of polynomials again. Hence we can apply a partial factorization to them as long as they are partially factorizable. This will lead us to *a Honor factorization of* $f \in \mathbb{C}[\boldsymbol{x}]$.

Algorithm 2.1.

- Input : $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$.
- Output : A Hornor factorization $\mathbb{HF}(g)$ of $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ and the number $\eta(\mathbb{HF}(g))$ of the multiplications to evaluate $g(\boldsymbol{x})$ using the Hornor factorization. Here $\mathbb{HF}(g)$ and $\eta(\mathbb{HF}(g))$ are generated recursively as shown below.

Step 1: If g is non-factorizable (*i.e.*, $\mathcal{Q}(g) = \emptyset$) then let $\mathbb{HF}(g) = g(\boldsymbol{x})$ and $\eta(\mathbb{HF}(g)) = \sum_{p \in P} \deg(\boldsymbol{\beta}_p)$. Otherwise go to Step 2.

Step 2: Choose a $Q \in \mathcal{Q}(g)$.

Step 3: Represent g as in (13).

Step 4: Let

$$\begin{aligned}
\mathbb{HF}(g) &= \boldsymbol{x}^{\boldsymbol{\gamma}(g, Q)}\mathbb{HF}(\varphi_L(\cdot; g, Q)) + \mathbb{HF}(\varphi_R(\cdot; g, Q)), \\
\eta(\mathbb{HF}(g)) &= \deg(\boldsymbol{\gamma}(g, Q)) + \eta(\mathbb{HF}(\varphi_L(\cdot; g, Q))) + \eta(\mathbb{HF}(\varphi_L(\cdot; g, Q))).
\end{aligned}$$

To generate a Hornor factorization of $f = (M, \boldsymbol{\alpha}_p)$ by applying Algorithm 2.1 to $f = (M, \boldsymbol{\alpha}_p)$, we need to specify how we choose a $Q \in \mathcal{Q}(g)$ at Step 2. Ideally, we want to find a minimum Hornor factorization, *i.e.*, a Hornor factorization $\mathbb{HF}(f)$ of $f = (M, \boldsymbol{\alpha}_p)$ which minimizes the number $\eta(\mathbb{HF}(f))$ of multiplications to evaluate $f(\boldsymbol{x})$ over all possible Hornor factorizations of $f = (M, \boldsymbol{\alpha}_p)$. This will be discussed in Section 3. As we can easily guess, the combinatorial explosion generally occurs in $\mathcal{Q}(g)$, *i.e.*, the number of candidates Q from $\mathcal{Q}(g)$ in Step 2 grows very rapidly as the number of $\boldsymbol{\beta}_p$ $(p \in P)$ increases and/or the degrees of $\boldsymbol{x}^{\boldsymbol{\beta}_p}$ $(p \in P)$ get larger. Because of this reason, an optimal Hornor factorization of $f = (M, \boldsymbol{\alpha}_p)$ is not tractable in such cases. Therefore we need some heuristic methods for choosing a $Q \in \mathcal{Q}(g)$ for Hornor factorizations with small numbers of multiplications, which we will discuss in Section 4.

Figure 3 illustrates an application of Algorithm 2.1 to the polynomial $f \in \mathbb{C}[\boldsymbol{x}]$ given in (6) in terms of a tree, which we will call *a Hornor tree*. The root node

$$
\begin{aligned}
&Q_0 = M = \{1, 2, 3, 4, 5, 6\}, \\
&g_0(\boldsymbol{x}) = c_1x_1x_2x_3x_4 + c_2x_2x_3x_4x_5 + c_3x_1x_3x_4x_5 + c_4x_1x_2x_4x_5 + c_5x_1x_2x_3x_5 + c_6
\end{aligned}
$$

of the tree is the original polynomial $f \in \mathbb{R}[\boldsymbol{x}]$ given in (6) itself, which is partially-factorizable. At Step 2, we take $Q_1 = \{1, 2, 3\} \in \mathcal{Q}(g_0)$ then we obtain the two nodes in the second level

$$
\begin{aligned}
&Q_1 = \{1, 2, 3\} \subseteq Q_0,\ \boldsymbol{\gamma}(g_0, Q_1) = (0, 0, 1, 1, 0),\ \boldsymbol{x}^{\boldsymbol{\gamma}(g_0, Q_1)} = x_3x_4, \\
&g_1(\boldsymbol{x}) = \varphi_L(\boldsymbol{x}; g_0, Q_1) = c_1x_1x_2 + c_2x_2x_5 + c_3x_1x_5, \\
&\text{ and} \\
&Q_2 = Q_0 \backslash Q_1 = \{4, 5, 6\},\ g_2(\boldsymbol{x}) = \varphi_R(\boldsymbol{x}, g_0, Q_1) = c_4x_1x_2x_4x_5 + c_5x_1x_2x_3x_5 + c_6.
\end{aligned}
$$

At the left node in the second level, we take $Q_3 = \{1, 2\} \subset \mathcal{Q}(g_1)$ and obtain two nodes in the third level:

$$
\begin{aligned}
&Q_3 = \{1, 2\} \subset Q_1,\ \boldsymbol{\gamma}(g_1, Q_3) = (0, 1, 0, 0, 0),\ \boldsymbol{x}^{\boldsymbol{\gamma}(g_1, Q_3)} = x_2, \\
&g_3(\boldsymbol{x}) = \varphi_L(\boldsymbol{x}, g_1, Q_3) = c_1x_1 + c_2x_5, \\
&\text{ and} \\
&Q_4 = Q_1 \backslash Q_3 = \{3\},\ g_4(\boldsymbol{x}) = \varphi_R(\boldsymbol{x}; g_1, Q_3) = c_3x_1x_5.
\end{aligned}
$$

Taking $Q_5 = \{4, 5\} \in \mathcal{G}(g_2)$ at the right node on the second level, we similarly obtain two nodes in the third level:

$$
\begin{aligned}
&Q_5 = \{4, 5\} \subset Q_2,\ \boldsymbol{\gamma}(g_2, Q_5) = (1, 1, 0, 0, 1),\ \boldsymbol{x}^{\boldsymbol{\gamma}(g_2, Q_5)} = x_1x_2x_5, \\
&g_5(\boldsymbol{x}) = \varphi_L(\boldsymbol{x}; g_2, Q_5) = c_4x_4 + c_5x_1x_3, \\
&Q_6 = Q_2 \backslash Q_5,\ g_6(\boldsymbol{x}) = \varphi_R(\boldsymbol{x}; g_2, Q_5) = c_6.
\end{aligned}
$$

Now the polynomials g_3, g_4, g_5 and g_6 at the leaf nodes are non-factorizable, so that this completes a Honor factorization of the polynomial f. The resulting Hornor factorization can be build up along the paths from these leaf nodes to the root node. First we observe that

$$
\begin{aligned}
\eta(\mathbb{HF}(g_3)) &= \text{t.deg}(g_3) = 2,\ \eta(\mathbb{HF}(g_4)) = \text{t.deg}(g_4) = 2, \\
\eta(\mathbb{HF}(g_5)) &= \text{t.deg}(g_5) = 2,\ \eta(\mathbb{HF}(g_6)) = \text{t.deg}(g_6) = 0.
\end{aligned}
\tag{14}
$$

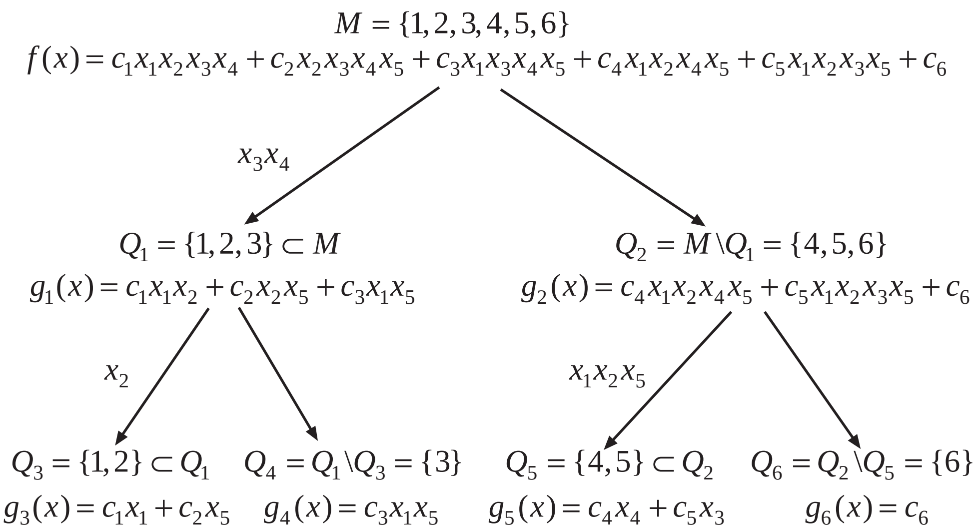

Figure 3: A Hornor factorization of the polynomial $f \in \mathbb{R}[\boldsymbol{x}]$ given in (6)

Then the polynomial g_1 in the second level is represented in terms of the polynomials of its child nodes, g_3 and g_4:

$$\begin{aligned}
g_1(\boldsymbol{x}) &= x_2g_3(\boldsymbol{x}) + g_4(\boldsymbol{x}) = x_2(c_1x_1 + c_2x_5) + c_3x_1x_5, \\
\eta(\text{HF}(g_1)) &= 1 + 2 + 2 = 5.
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
g_2(\boldsymbol{x}) &= x_1x_2x_5g_5(\boldsymbol{x}) + g_6(\boldsymbol{x}) \\
&= x_1x_2x_5(c_4x_4 + c_5x_3) + c_6, \\
\eta(\text{HF}(g_2)) &= 3 + 2 + 0 = 5,
\end{aligned}$$

and finally

$$\begin{aligned}
f(\boldsymbol{x}) &= g_0(\boldsymbol{x}) \\
&= x_3x_4g_1(\boldsymbol{x}) + g_2(\boldsymbol{x}) \\
&= x_3x_4(x_2(c_1x_1 + c_2x_5) + c_3x_1x_5) + x_1x_2x_5(c_4x_4 + c_5x_3) + c_6; \text{ hence} \\
\text{HF}(f) &= x_3x_4(x_2(c_1x_1 + c_2x_5) + c_3x_1x_5) + x_1x_2x_5(c_4x_4 + c_5x_3) + c_6, \\
\eta(\text{HF}(f)) &= 5 + 5 + 2 = 12.
\end{aligned}$$

2.5 Computation of the function value $f(\boldsymbol{x})$

Computation of the function value $f(\boldsymbol{x})$ is carried out in a similar way as we build a Hornor factorization above using a Hornor tree. We continue to use the same example above whose Hornor tree is given in Figure 3. First we compute the values of the polynomials g_3, g_4, g_5 and g_6 at the leaf nodes. Then, we compute the values of $x_2g_3(\boldsymbol{x}) + g_4(\boldsymbol{x})$ and $g_2(\boldsymbol{x}) = x_1x_2x_5g_5(\boldsymbol{x}) + g_6(\boldsymbol{x})$. Finally, we obtain that $f(\boldsymbol{x}) = g_0(\boldsymbol{x}) = x_3x_4g_1(\boldsymbol{x}) + g_2(\boldsymbol{x})$.

3 The minimum number of multiplications over all Hornor factorizations

3.1 Recursive formula

For every $g \in \mathbb{F}[\boldsymbol{x}, f]$, let $\nu(g)$ denote the minimum number of multiplications to evaluate $g(\boldsymbol{x})$ over all possible Honor factorizations. Suppose that $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ with $k = \min P$. Then

$$\nu(g) \;=\; \begin{cases} \text{t.deg}(g) \text{ if } g = (P, \boldsymbol{\beta}_p) \text{ is non-factorizable or } \mathcal{Q}(g) = \emptyset, \\ \min\left\{\nu(Q, \boldsymbol{\beta}_p) + \nu(P \backslash Q, \boldsymbol{\beta}_p) : k \in Q \subseteq P\right\} \text{ otherwise.} \end{cases} \tag{15}$$

We now focus our attention to the latter case above where we have

$$\nu(g) \;=\; \min\left\{\nu(Q, \boldsymbol{\beta}_p) + \nu(P \backslash Q, \boldsymbol{\beta}_p) : k \in Q \subseteq P\right\}. \tag{16}$$

If $\boldsymbol{\gamma}(g, P) \neq \emptyset$, then

$$g(\boldsymbol{x}) \;=\; \boldsymbol{x}^{\boldsymbol{\gamma}(g,P)} \left(\sum_{p \in P} c_p \boldsymbol{x}^{\boldsymbol{\beta}_p - \boldsymbol{\gamma}(g,P)} \right);$$

hence

$$\nu(g) \;=\; \deg(\boldsymbol{\gamma}(g, P)) + \nu(P, \boldsymbol{\beta}_p - \boldsymbol{\beta}(g, P)).$$

Thus the minimum is attained with $Q = P$. Suppose now that the minimum is attained at some $Q = Q^*$ such that $k \in Q^* \subset P$ and $Q^* \neq P$. Then

$$\nu(g) \;=\; \nu(Q^*, \boldsymbol{\beta}_p) + \nu(P \backslash Q^*, \boldsymbol{\beta}_p).$$

Then we have either

(i) $Q^* = \{k\}$.

(ii) $k \in Q^*$, $\#Q^* \geq 2$ and $\boldsymbol{\gamma}(g, Q^*) \neq \mathbf{0}$.

(iii) $k \in Q^*$, $\#Q^* \geq 2$ and $\boldsymbol{\gamma}(g, Q^*) = \mathbf{0}$.

In case (iii), there exists a nonempty proper subset $\hat{Q}$ of Q^* with $k \in \hat{Q}$ such that

$$\nu(Q^*, \boldsymbol{\beta}_p) \;=\; \nu(\hat{Q}, \boldsymbol{\beta}_p) + \nu(Q^* \backslash \hat{Q}, \boldsymbol{\beta}_p).$$

Hence we have

$$\begin{aligned} \nu(g) \;&=\; \nu(\hat{Q}, \boldsymbol{\beta}_p) + \nu(Q^* \backslash \hat{Q}, \boldsymbol{\beta}_p) + \nu(P \backslash Q^*, \boldsymbol{\beta}_p) \\ &=\; \nu(\hat{Q}, \boldsymbol{\beta}_p) + \nu(P \backslash \hat{Q}, \boldsymbol{\beta}_p). \end{aligned}$$

Redefine $Q^* = \hat{Q}$. If case (iii) still holds for this Q^*, we can continue to apply the same argument till either (i) or (ii) holds. Therefore we can impose an additional condition

$\boldsymbol{\gamma}(g,Q) \neq \mathbf{0}$ (or $Q \in \mathcal{Q}(g)$) or $Q = \{k\}$ in evaluating the minimum on the right hand side of (16). Therefore we can replace (15) by

$$\nu(g) \quad = \quad \begin{cases} \text{t.deg}(g) \text{ if } g = (P, \boldsymbol{\beta}_p) \text{ is non-factorizable or } \mathcal{Q}(g) = \emptyset, \\ \min \left\{ \nu(Q, \boldsymbol{\beta}_p) + \nu(P \backslash Q, \boldsymbol{\beta}_p) : \begin{array}{c} k \in Q \in \mathcal{Q}(g), \\ \text{or } Q = \{k\} \end{array} \right\} \text{ otherwise.} \end{cases} \tag{17}$$

As an example, let us apply the formula (17) to the polynomial $g = (P, \boldsymbol{\beta}_p)$ given in (5). Recall that (12) holds. Since $k = \min P = 1$ and $\mathcal{Q}(g) = \{\{1,3\},\ \{1,4\}, \{3,4\}\}$, we have that

$$\begin{aligned}
\nu(g) &= \min \big\{ \nu(\{1,3\}, \boldsymbol{\beta}_p) + \nu(\{4\}, \boldsymbol{\beta}_p), \\
&\qquad\qquad \nu(\{1,4\}, \boldsymbol{\beta}_p) + \nu(\{3\}, \boldsymbol{\beta}_p), \\
&\qquad\qquad \nu(\{1\}, \boldsymbol{\beta}_p) + \nu(\{3,4\}, \boldsymbol{\beta}_p) \big\}, \\
\nu(\{1,3\}, \boldsymbol{\beta}_p) &= \nu(c_1x_1x_2x_3 + c_3x_1x_3x_5) = \nu(x_1x_5(c_1x_2 + c_3x_5)) \\
&= 2 + \nu(c_1x_2 + c_3x_5) = 2 + 2 = 4, \\
\nu(\{4\}, \boldsymbol{\beta}_p) &= \nu(c_4x_2x_5) = 2, \\
\nu(\{1,4\}, \boldsymbol{\beta}_p) &= \nu(c_1x_1x_2x_3 + c_4x_2x_5) = \nu(x_2(c_1x_1x_3 + c_4x_5)) \\
&= 1 + \nu(c_1x_1x_3 + c_4x_5) = 1 + 3 = 4, \\
\nu(\{3\}, \boldsymbol{\beta}_p) &= \nu(c_3x_1x_3x_5) = 3, \\
\nu(\{1\}, \boldsymbol{\beta}_p) &= \nu(c_1x_1x_2x_3) = 3, \\
\nu(\{3,4\}, \boldsymbol{\beta}_p) &= \nu(c_3x_1x_3x_5 + c_4x_2x_5) = \nu(x_5(c_3x_1x_3 + c_4x_2)) \\
&= 1 + \nu(c_3x_1x_3 + c_4x_2) = 1 + 3 = 4.
\end{aligned}$$

Hence

$$\nu(g) \quad = \quad \min\{4+2,\ 4+3,\ 3+4\} = 6.$$

Using the formula (17) recursively, we could compute the minimum number of multiplications over all possible Hornor factorizations of a polynomial $f = (M, \boldsymbol{\alpha}_p) \in \mathbb{C}[\boldsymbol{x}]$ of the form (2). In the next subsection, we present lower bounds for the number of multiplications to evaluate a polynomial $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$. Incorporating the lower bounds into the recursive formula (17), we can improve its efficiency to compute the minimum number of multiplications in evaluating the polynomial $f = (M, \boldsymbol{\alpha}_p)$.

3.2 Lower bounds for the number of multiplications in Hornor factorizations

Suppose that $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ and $k = \min P$. We introduce a lower bound $\lambda_1(g)$ for the number of multiplications to evaluate g over all possible Honor factorizations:

$$\lambda_1(g) \quad = \quad \lambda_1(P, \boldsymbol{\beta}_p) = \sum_{i=1}^{n} \min \left\{ [\boldsymbol{\beta}_p]_i : p \in P \right\}.$$

Note that to evaluate $g(\boldsymbol{x})$ x_i is multiplied at least $\min \left\{ [\boldsymbol{\beta}_p]_i : p \in P \right\}$ times for every $i = 1, 2, \ldots, n$. In the case of $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ given in (7), we see that

$$\lambda_1(g) \quad = \quad \lambda_1(c_1x_1x_2x_3 + c_3x_1x_3x_5 + c_4x_2x_5) = 5.$$

In general, $\lambda_1(P,\boldsymbol{\beta}_p) \leq \nu(P,\boldsymbol{\beta}_p) \leq \mathrm{t.deg}(P,\boldsymbol{\beta}_p)$. When $g = (P,\boldsymbol{\beta}_p)$ is non-factorizable or $\mathcal{Q}(g) = \emptyset$, we see that $\lambda_1(g) = \nu(g) = \mathrm{t.deg}(P,\boldsymbol{\beta}_p)$. Hence the lower bound $\lambda_1(g)$ for $\nu(g)$ is tight in this case. Otherwise we know from (17) that

$$\begin{aligned}\nu(g) &= \min\left\{\nu(Q,\boldsymbol{\beta}_p) + \nu(P\backslash Q,\boldsymbol{\beta}_p) : \begin{array}{l} k \in Q \in \mathcal{Q}(g), \\ \text{or } Q = \{k\} \end{array}\right\} \\ &\geq \min\left\{\lambda_1(Q,\boldsymbol{\beta}_p) + \lambda_1(P\backslash Q,\boldsymbol{\beta}_p) : \begin{array}{l} k \in Q \in \mathcal{Q}(g), \\ \text{or } Q = \{k\} \end{array}\right\}\end{aligned}$$

Therefore we can define a better lower bound $\lambda_2(g) = \lambda(((P,\boldsymbol{\beta}_p))$ as follows.

$$\lambda_2(g) = \begin{cases} \mathrm{t.deg}(g) \ \text{ if } g = (P,\boldsymbol{\beta}_p) \text{ is non-factorizable or } \mathcal{Q}(g) = \emptyset, \\ \min\left\{\lambda_1(Q,\boldsymbol{\beta}_p) + \lambda_1(P\backslash Q,\boldsymbol{\beta}_p) : \begin{array}{l} k \in Q \in \mathcal{Q}(g), \\ \text{or } Q = \{k\} \end{array}\right\} \text{ otherwise.} \end{cases}$$

In the case of $g = (P,\boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ given in (7), we know that $P = \{1,3,4\}$, $\mathcal{Q}(g) = \{\{1,3\},\ \{1,4\},\ \{3,4\}\}$ and $k = 1$ (recall (12)). Hence

$$\begin{aligned}\lambda_1(g) &= 3, \\ \lambda_2(g) &= \min\{\lambda_1(\{1,3\},\boldsymbol{\beta}_p) + \lambda_1(\{4\},\boldsymbol{\beta}_p), \\ &\qquad \lambda_1(\{1,4\},\boldsymbol{\beta}_p) + \lambda_1(\{3\},\boldsymbol{\beta}_p), \\ &\qquad \lambda_1(\{1\},\boldsymbol{\beta}_p) + +\lambda_1(\{3,4\},\boldsymbol{\beta}_p)\} \\ &= \min\{\lambda_1(c_1x_1x_2x_3 + c_3x_1x_3x_5) + \lambda_1(c_4x_2x_5), \\ &\qquad \lambda_1(c_1x_1x_2x_3 + c_4x_2x_5) + \lambda_1(c_3x_1x_3x_5), \\ &\qquad \lambda_1(c_1x_1x_2x_3) + +\lambda_1(c_4x_2x_5 + c_3x_1x_3x_5)\} \\ &= \min\{4+2,\ 4+2,\ 3+3\} = 6.\end{aligned}$$

3.3 Saving the work to compute $\nu(g)$ by using its lower bounds

Suppose that $g = (P,\beta_p) \in \mathbb{F}[\boldsymbol{x}, f]$ and $\mathcal{Q}(g) \neq \emptyset$. In order to compute $\nu(g)$, we generate a family of subsets of P,

$$\{Q \subseteq P : k \in Q \in \mathcal{Q}(g) \ \text{ or } Q = \{k\}\},$$

where $k = \min P$. Let the members of this family be

$$Q_1,\ Q_2,\ \ldots,\ Q_{s-1} \in \mathcal{Q}(g) \ \text{ and } Q_s = \{k\}.$$

Then

$$\nu(g) = \min\{\nu(Q_j,\boldsymbol{\beta}_p) + \nu(P\backslash Q_j,\boldsymbol{\beta}_p) : j = 1,2,\ldots,s\}$$

Hence we observe that, for $j = 1,2,\ldots,s-1$,

$$\begin{aligned}&\nu(Q_j,\boldsymbol{\beta}_p) + \nu(P\backslash Q_j,\boldsymbol{\beta}_p) \\ &\quad = \deg(\boldsymbol{\gamma}(g,Q_j)) + \nu(Q_j,\boldsymbol{\beta}_p - \boldsymbol{\gamma}(g,Q_j)) + \nu(P\backslash Q_j,\boldsymbol{\beta}_p) \\ &\quad \geq \deg(\boldsymbol{\gamma}(g,Q_j)) + \lambda(Q_j,\boldsymbol{\beta}_p - \boldsymbol{\gamma}(g,Q_j)) + \lambda(P\backslash Q_j,\boldsymbol{\beta}_p),\end{aligned}$$

and that, for $j = s$,

$$\begin{aligned} \nu(Q_r, \boldsymbol{\beta}_p)) + \nu((P \backslash Q_r, \boldsymbol{\beta}_p) &= \deg(\boldsymbol{\beta}_k) + \nu(P \backslash \{k\}, \boldsymbol{\beta}_p) \\ &\geq \deg(\boldsymbol{\beta}_k) + \lambda(P \backslash \{k\}, \boldsymbol{\beta}_p). \end{aligned}$$

Here $\lambda(h)$ denotes either of the lower bounds $\lambda_1(h)$ and $\lambda_2(h)$ for the minimum number $\nu(h)$ of multiplications to evaluate $h \in \mathbb{F}[\boldsymbol{x}, f]$.

Now suppose that we have computed

$$\nu_j \ = \ \nu(Q_j, \boldsymbol{\beta}_p) + \nu(P \backslash Q_j, \boldsymbol{\beta}_p) \ (j = 1, 2, \ldots, r)$$

for some $r \leq s - 1$. Let $\nu^* = \min\{\nu_j : j = 1, 2, \ldots, r\}$. Before computing

$$\nu_{r+1} \ = \ \nu(Q_{r+1}, \boldsymbol{\beta}_p) + \nu(P \backslash Q_{r+1}, \boldsymbol{\beta}_p),$$

we compute

$$\hat{\lambda} \ = \ \begin{cases} \deg(\boldsymbol{\gamma}(g, Q_{r+1})) + \lambda(Q_{r+1}, \boldsymbol{\beta}_p - \boldsymbol{\gamma}(g, Q_{r+1})) + \lambda(P \backslash Q_{r+1}, \boldsymbol{\beta}_p) \\ \qquad\qquad\qquad \text{if } r + 1 < s, \\ \deg(\boldsymbol{\beta}_k) + \lambda(P \backslash \{k\}, \boldsymbol{\beta}_p) \text{ if } r + 1 = s. \end{cases}$$

If $\nu^* \leq \hat{\lambda}$, then we know from the discussion above that $\nu^* \leq \hat{\lambda} \leq \nu_{r+1}$. Hence ν_{r+1} can not improve the currently known best number ν^* of multiplications to evaluate $g(\boldsymbol{x})$, so that we can skip the computation of ν_{r+1}. This saves the work to compute $\nu(g)$ because the computation of λ is less expensive than that of ν_{r+1} in general.

In Section 6, we show through some numerical results that how effectively the lower bounds λ_1 and λ_2 save the number of recursive calls of ν to compute minimal Hornor factorizations of polynomials.

4 Heuristic methods

As we mentioned, in order to generate a Hornor factorization of a given polynomial $f = (M, \boldsymbol{\alpha}_p)$ of the form (2) by applying Algorithm 2.1, we need to choose a Q from $\mathcal{Q}(g)$ at Step 2 of Algorithm 2.1. In this section, we present three heuristic methods for choosing Q from $\mathcal{Q}(g)$ there in Sections 4.1, 4.2 and 4.3, respectively. Numerical results on the heuristic methods proposed in this section in comparison to the recursive formula (15) incorporated with the lower bound λ_2 will be reported in Section 6.

4.1 Heuristic method 1 using the best upper bound

The first method utilizes upper bounds for the minimum number $\nu(g)$ of multiplications to evaluate $g(\boldsymbol{x})$, which we can compute with less cost than $\nu(g)$ itself. Suppose that $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ and $k = \min P$. Let

$$\begin{aligned} \mu_1(g) &= \mu_1(P, \boldsymbol{\beta}_p) \\ &= \begin{cases} \text{t.deg}(P, \boldsymbol{\beta}_p) \text{ if } \boldsymbol{\gamma}(P, g) = \mathbf{0} \text{ or } \#P = 1, \\ \deg(\boldsymbol{\gamma}(g, P)) + \mu_1(P, \boldsymbol{\beta}_p - \boldsymbol{\gamma}(g, P)) \text{ otherwise.} \end{cases} \end{aligned}$$

Note that $\boldsymbol{\gamma}(P, \boldsymbol{\beta}_p - \boldsymbol{\gamma}(g, P)) = \mathbf{0}$ in the latter case; hence

$$\mu_1(g) = \deg(\boldsymbol{\gamma}(g, P)) + \text{t.deg}(P, \boldsymbol{\beta}_p - \boldsymbol{\gamma}(g, P)) < \text{t.deg}(P, \boldsymbol{\beta}_p).$$

In general, $\nu(g) \leq \mu_1(g)$. If $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ is non-factorizable, we see that $\lambda_1(g) = \nu(g) = \mu_1(g)$.

Taking the recursive formula (17) into account as in Section 3.2, we can strengthen the upper bound $\mu_1(g)$, and define a better upper bound $\mu_2(g)$ for $\nu(g)$:

$$\mu_2(g) \quad = \quad \begin{cases} \text{t.deg}(P, \boldsymbol{\beta}_p) \text{ if } g = (P, \boldsymbol{\beta}_p) \text{ is non-factorizable or } \mathcal{Q}(g) = \emptyset, \\ \min \left\{ \mu_1(Q, \boldsymbol{\beta}_p) + \mu_1(P \backslash Q, \boldsymbol{\beta}_p) : \begin{array}{l} k \in Q \in \mathcal{Q}(g), \\ \text{or } Q = \{k\} \end{array} \right\} \text{ otherwise.} \end{cases}$$

Here $k = \min P$.

In the case of $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ given in (7), we see that

$$\begin{aligned}
\mu_1(g) &= \mu_1(c_1x_1x_2x_3 + c_3x_1x_3x_5 + c_4x_2x_5) = 8, \\
\mu_2(g) &= \min \{\mu_1(\{1,3\}, \boldsymbol{\beta}_p) + \mu_1(\{4\}, \boldsymbol{\beta}_p), \\
&\qquad \mu_1(\{1,4\}, \boldsymbol{\beta}_p) + \mu_1(\{3\}, \boldsymbol{\beta}_p), \\
&\qquad \mu_1(\{1\}, \boldsymbol{\beta}_p) + +\mu_1(\{3,4\}, \boldsymbol{\beta}_p)\} \\
&= \min \{\mu_1(c_1x_1x_2x_3 + c_3x_1x_3x_5) + \mu_1(c_4x_2x_5), \\
&\qquad \mu_1(c_1x_1x_2x_3 + c_4x_2x_5) + \mu_1(c_3x_1x_3x_5), \\
&\qquad \mu_1(c_1x_1x_2x_3) + +\mu_1(c_3x_1x_3x_5 + c_4x_2x_5)\} \\
&= \min \{\mu_1(x_1x_3(c_1x_2 + c_3x_5)) + \mu_1(c_4x_2x_5), \\
&\qquad \mu_1(x_2(c_1x_1x_3 + c_4x_5)) + \mu_1(c_3x_1x_3x_5), \\
&\qquad \mu_1(c_1x_1x_2x_3) + +\mu_1(x_5(c_3x_1x_3 + c_4x_2))\} \\
&= \min\{4+2,\ 4+3,\ 3+4\} = 6.
\end{aligned}$$

Now we are ready to describe Heuristic method 1.

Algorithm 4.1. (Heuristic method 1)

- Input $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$. Here we assume that $g = (P, \boldsymbol{\beta}_p)$ is partially factorizable or $\mathcal{Q}(g) \neq \emptyset$.
- Output $Q \in \mathcal{Q}(g)$ for Step 2 of Algorithm 2.1.

Step 1: If $\boldsymbol{\gamma}(P, \boldsymbol{\beta}_p) \neq \mathbf{0}$ then output $Q = P$.

Step 2: Othewise, output an optimal solution Q of the problem

$$\begin{array}{ll} \text{minimize} & \mu(Q, \boldsymbol{\beta}_p) + \mu(P \backslash Q, \boldsymbol{\beta}_p) \\ \text{subject to} & k \in Q \in \mathcal{Q}(g) \text{ or } Q = \{k\}. \end{array}$$

Here $k = \min P$ and μ denotes either μ_1 or μ_2.

The heuristic method above is less expensive than the computation of the minimum number $\nu(g)$ of multiplications to evaluate $g(\boldsymbol{x})$. But if we employ $\mu = \mu_2$, the minimization problem requires to compute $\mathcal{Q}(g)$ as in the computation of $\nu(g)$. Therefore Heuristic method 1 with the use of $\mu = \mu_2$ rapidly becomes more expensive to excecute as the number of $\boldsymbol{\beta}_p$ $(p \in P)$ increases and/or the degrees of $\boldsymbol{x}^{\boldsymbol{\beta}_p}$ $(p \in P)$ get larger.

4.2 Heuristic method 2 taking account of a certain similarity among monomials

In this subsection and the next, we propose less expensive heuristic methods than the one presented in the previous subsection for computing Hornor factiorizations with small numbers of multiplications to compute $f = (M, \boldsymbol{\alpha}_p)$. In the method described below in this subsection, we utilize $\deg(\boldsymbol{\gamma}(Q, g))$ to represent a similarity for each $Q \in \mathcal{Q}(g)$. First we choose one of the most similar pairs from P, say q and q', and set $Q = \{q\}$. Then we add add a $q \in P \backslash Q$ by one by one as long as $\mu_1(Q \cup \{q\}, g) + \mu_1(P \backslash (Q \cup \{q\}), g)$ gets smaller than $\mu_1(Q, g) + \mu_1(P \backslash Q, g)$.

Algorithm 4.2. (Heuristic method 2)

- Input $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$. Here we assume that $g = (P, \boldsymbol{\beta}_p)$ is partially factorizable or $\mathcal{Q}(g) \neq \emptyset$.
- Output $Q \in \mathcal{Q}(g)$ for Step 2 of Algorithm 2.1.

Step 1: If $\boldsymbol{\gamma}(P, \boldsymbol{\beta}_p) \neq \mathbf{0}$ then output $Q = P$.

Step 2: Choose a pair of distinct $q,\ q' \in P$ such that

$$\deg(\boldsymbol{\gamma}(\{q, q'\}, \boldsymbol{\beta}_p) = \max\{\deg(\boldsymbol{\gamma}(\{r, r'\}, \boldsymbol{\beta}_p) : r, r' \in P,\ r \neq r'\}.$$

Let $Q = \{q\}$.

Step 3: If $Q = P$ then output Q. Otherwise let $q \in P \backslash Q$ be such that

$$\deg(\boldsymbol{\gamma}(Q \cup \{q\}, \boldsymbol{\beta}_p) = \max\{\deg(\boldsymbol{\gamma}(Q \cup \{r\}, \boldsymbol{\beta}_p) : r, \in P,\ r \in P \backslash Q\}.$$

Step 4: If

$$\mu_1(Q, \boldsymbol{\beta}_p) + \mu_1(P \backslash Q, \boldsymbol{\beta}_p) > \mu_1(Q \cup \{q\}, \boldsymbol{\beta}_p) + \mu_1(P \backslash (Q \cup \{q\}), \boldsymbol{\beta}_p),$$

then let $Q = Q \cup \{q\}$, and go to Step 3. Otherwise output Q.

Now let us apply Heuristic method 2 to the case of $g = (P, \boldsymbol{\beta}_p) \in \mathbb{F}[\boldsymbol{x}, f]$ given in (7). In this case, we have that

$$\begin{aligned} \boldsymbol{\gamma}(\{1, 3, 4\}, \boldsymbol{\beta}_p) &= \mathbf{0}, \\ \deg(\boldsymbol{\gamma}(\{1, 3\}, \boldsymbol{\beta}_p)) &= 2 > 1 = \deg(\boldsymbol{\gamma}(\{1, 4\}, \boldsymbol{\beta}_p)) = \deg(\boldsymbol{\gamma}(\{3, 4\}, \boldsymbol{\beta}_p)), \end{aligned}$$

so that we set $Q = \{1\}$ at Step 2. Then we observe that

$$\deg(\{1\} \cup \{3\}, \boldsymbol{\beta}_p) = 2 > 1 \deg(\{1\} \cup \{4\}, \boldsymbol{\beta}_p)$$

at Step 3, and

$$\begin{array}{rcl}
\mu_1(\{1\},\boldsymbol{\beta}_p)+\mu_1(\{3,4\},\boldsymbol{\beta}_p) & = & \mu_1(c_1x_1x_2x_3)+\mu_1(x_5(c_3x_1x_3+c_4x_2)) \\
& = & 3+4=7, \\
\mu_1(\{1\}\cup\{3\},\boldsymbol{\beta}_p)+\mu_1(\{4\},\boldsymbol{\beta}_p) & = & \mu_1(x_1x_3(c_1x_2+c_3x_5))+\mu_1(c_4x_2x_5) \\
& = & 4+2=6;\ \text{hence} \\
\mu_1(\{1\},\boldsymbol{\beta}_p)+\mu_1(\{3,4\},\boldsymbol{\beta}_p) & > & \mu_1(\{1\}\cup\{3\},\boldsymbol{\beta}_p)+\mu_1(\{4\},\boldsymbol{\beta}_p)
\end{array}$$

at Step 4. Thus we update $Q=\{1\}$ to $Q=\{1,3\}$, and go back to Step 3. Now $q=4$ is uniquely chosen at Step 3, but we see that

$$\mu_1(\{1,3\}\cup\{4\},\boldsymbol{\beta}_p) \;=\; \deg(\{1,3,4\},\boldsymbol{\beta}_p)=8>6=\mu_1(\{1,3\},\boldsymbol{\beta}_p)+\mu_1(\{4\},\boldsymbol{\beta}_p).$$

Therefore, the method outputs $Q=\{1,3\}$.

4.3 Heuristic method 3 taking account of the number of factorized monomials

We now focus our attention to the number of elements in $Q\in\mathcal{Q}(g)$, and choose $Q\in\mathcal{Q}(g)$ having the maximum number of elements among members of $\mathcal{Q}(g)$, which is computed as follows:

$$\left.\begin{array}{l}
\#\{p\in P:[\boldsymbol{\beta}_p]_i\geq 1\}=\max\limits_{1\leq j\leq n}\#\{p\in P:[\boldsymbol{\beta}_p]_j\geq 1\}, \\
Q=\{p\in P:[\boldsymbol{\beta}_p]_i\geq 1\}.
\end{array}\right\} \tag{18}$$

Algorithm 4.3. (Heuristic method 3)

- Input $g=(P,\boldsymbol{\beta}_p)\in\mathbb{F}[\boldsymbol{x},f]$. Here we assume that $g=(P,\boldsymbol{\beta}_p)$ is partially factorizable or $\mathcal{Q}(g)\neq\emptyset$.
- Output $Q\in\mathcal{Q}(g)$ for Step 2 of Algorithm 2.1.

Step 1: If $\boldsymbol{\gamma}(P,\boldsymbol{\beta}_p)\neq\mathbf{0}$ then output $Q=P$.

Step 2: Otherwise, output $Q\in\mathcal{Q}(g)$ determined by (18).

If we apply Heuristic method 3 to the case of $g=(P,\boldsymbol{\beta}_p)\in\mathbb{F}[\boldsymbol{x},f]$ given in (7), w have that

$$\begin{array}{rcl}
\boldsymbol{\gamma}(\{1,3,4\},\boldsymbol{\beta}_p) & = & \mathbf{0}, \\
\#\{p\in P:[\boldsymbol{\beta}_p]_i\geq 1\} & = & 2\ (i=1,2,3,5), \\
\#\{p\in P:[\boldsymbol{\beta}_p]_4\geq 1\} & = & 0.
\end{array}$$

Hence, either of

$$\begin{array}{rcl}
\{1,3\} & = & \{p\in P:[\boldsymbol{\beta}_p]_1\geq 1\}=\{p\in P:[\boldsymbol{\beta}_p]_3\geq 1\}, \\
\{1,4\} & = & \{p\in P:[\boldsymbol{\beta}_p]_2\geq 1\}, \\
\{3,4\} & = & \{p\in P:[\boldsymbol{\beta}_p]_5\geq 1\}
\end{array}$$

is chosen for Q at Step 2.

5 Evaluation of a system of polynomials and their partial derivatives

We now present how efficiently we evaluate a system of polynomials and their partial derivatives. Here Hornor factorizations for a single polynomial discussed so far serve as a main tool. Consider a system of polynomials

$$\boldsymbol{f}(\boldsymbol{x}) \;=\; (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})),\; f_j \in \mathbb{C}[\boldsymbol{x}] \; (j = 1, 2, \ldots, m). \tag{19}$$

When we are concerned with a homotopy function of the form (1), $m = n$ and each $f_j(\boldsymbol{x})$ is corresponding to $h_j(\boldsymbol{x}, t)$ or $\partial h_j(\boldsymbol{x}, t)/\partial t$ for some fixed $t \in [0, 1]$. We assume that for each $j = 1, 2, \ldots, m$, a Hornor factorization of $f_j(\boldsymbol{x})$ with the minimum (or a small) number of multiplications has been already computed. Given a j, we first discuss how we evaluate partial derivatives $\partial f_j(\boldsymbol{x})/\partial x_i$ $(i = 1, 2, \ldots, n)$ using the Hornor factorization of $f_j(\boldsymbol{x})$ in Section 5.1, and then present a heuristic method to evaluate the collection of monomials which appear in Hornor factorizations of the polynomials $f_j(\boldsymbol{x})$ $(j = 1, 2, \ldots, m)$ and in the evaluation of their partial derivatives in Section 5.2.

5.1 Computation of values of partial derivatives of $f_j(\boldsymbol{x})$

For simplicity of notation, let $f = f_j \in \mathbb{C}[\boldsymbol{x}]$ for an arbitrary fixd $j \in \{1, 2, \ldots, m\}$. Once we have build up a Hornor factorization together with a Hornor tree of a polynomial f, values of partial derivatives of the polynomial is carried out by applying a method similar to the forward-mode automatic differentialtion (for example, see [14]). By using the chain rule, we represent the partial derivative $\partial g(\boldsymbol{x})/\partial x_i$ of a polynomial g of the form (13) as

$$\begin{aligned}
\frac{\partial g(\boldsymbol{x})}{\partial x_i} &= \frac{\partial \left(\boldsymbol{x}^{\boldsymbol{\gamma}(g,\,Q_1)}\varphi_L(\boldsymbol{x}; g, Q_1) + \varphi_R(\boldsymbol{x}; g, Q_1)\right)}{\partial x_i} \\
&= \frac{\partial \boldsymbol{x}^{\boldsymbol{\gamma}(g,\,Q_1)}}{\partial x_i}\varphi_L(\boldsymbol{x}; g, Q_1) + \boldsymbol{x}^{\boldsymbol{\gamma}(g,\,Q_1)}\frac{\partial \varphi_L(\boldsymbol{x}; g, Q_1)}{\partial x_i} \\
&\quad + \frac{\partial \varphi_R(\boldsymbol{x}; g, Q_1)}{\partial x_i}.
\end{aligned} \tag{20}$$

We then apply this formula to the partial derivative of the polynomial of each node from the leaves to the root recursively. In the case of the Hornor tree given in Figure 3, we first compute $\partial g_p(\boldsymbol{x})/\partial x_i$ $(i = 3, 4, 5, 6)$ at the leaf nodes in the third level. Then, applying the formula above, we compute $\partial g_1(\boldsymbol{x})/\partial x_i$, $\partial g_2(\boldsymbol{x})/\partial x_i$ and $\partial f(\boldsymbol{x})/\partial x_i = \partial g_0(\boldsymbol{x})/\partial x_i$.

5.2 Computation of monomials

When we explained how to compute the value of $f(\boldsymbol{x})$ in Section 2.5 and how to compute partial derivatives in Section 5.1, we assumed that the value of a monomial was computed independently from the value of another monomial. In the case of the Hornor tree given in

Figure 3, we needed the values of

$$\begin{array}{rl} x_1x_5 & \text{to compute } g_4(\boldsymbol{x}) = c_3x_1x_5, \\ x_1x_2x_5 & \text{to compute } g_2(\boldsymbol{x}) = x_1x_2x_5g_5(\boldsymbol{x}) + g_6(\boldsymbol{x}), \\ x_3x_4 & \text{to compute } g_0(\boldsymbol{x}) = x_3x_4g_1(\boldsymbol{x}) + g_2, \\ \dfrac{\partial(x_1x_2x_5)}{\partial x_i} & \text{to compute } \dfrac{\partial g_2(\boldsymbol{x})}{\partial x_i} \ (i = 1, 2, 5). \end{array}$$

The problem here is how we save the multiplications to evaluate all monomials. In this example, the answer is simple. We first compute $x_1x_2,\ x_1x_5,\ x_2x_5$ and x_3x_4, and then $x_1x_2x_5$ by multiplying x_1 and x_2x_5 or by multiplying x_1x_2 and x_5. In general, however, minimizing the number of multiplications to evaluate a set of given monomials is a complicated and difficult problem, and we will present a heuristic method for this problem below.

Let $\mathcal{B}$ be a nonempty finite subset of $\mathbb{Z}^n_+\backslash\{\mathbf{0}\}$, and let $\{\boldsymbol{x}^{\boldsymbol{\beta}} : \boldsymbol{\beta} \in \mathcal{B}\}$ be a set of monomials to be evaluated. For simplicity of notation, we will identify the set of monomials $\{\boldsymbol{x}^{\boldsymbol{\beta}} : \boldsymbol{\beta} \in \mathcal{B}\}$ with the set of their supports $\mathcal{B}$. We assume that $\mathcal{B}$ contains the n-dimensional unit coordinate vectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n$ or $x_1, x_2, \ldots, x_n$. Suppose that we are given a value $\bar{\boldsymbol{x}}$ for the variable vector $\boldsymbol{x} \in \mathbb{C}^n$. Then the values of monomials $\boldsymbol{x}^{\boldsymbol{\beta}} \in \mathcal{B}$ with degree 1 are decided, *i.e.*, $x_i = \bar{x}_i$ $(i = 1, 2, \ldots, n)$. Then we will compute the value of each higher degree monomial as the product of some two lower degree monomials recursively. For example, the value $\bar{x}_1\bar{x}_2$ is computed by multiplying $\bar{x}_1$ and $\bar{x}_2$, the value $\bar{x}_1\bar{x}_2\bar{x}_4$ by multiplying $\bar{x}_1\bar{x}_2$ and $\bar{x}_4$, the value $\bar{x}_1^2\bar{x}_2^2\bar{x}_4$ by multiplying $\bar{x}_1\bar{x}_2$ and $\bar{x}_1\bar{x}_2\bar{x}_4$, and so on. In this example, we have assumed that the monomials x_1x_2, $x_1x_2x_4$ and $x_1^2x_2^2x_4$ are members of $\mathcal{B}$ to be computed. If either of them is not a member of $\mathcal{B}$, we need to add it to $\mathcal{B}$.

Now we describe technical details of the method outlined above. To each $\boldsymbol{\beta} \in \mathcal{B}$, we will attach a positive integer $\kappa(\boldsymbol{\beta})$ and a set $\mathcal{C}(\boldsymbol{\beta})$ of two children of $\boldsymbol{\beta}$ such that

- if $\kappa(\boldsymbol{\beta}) < \kappa(\boldsymbol{\beta}')$ then $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}'}$ is computed in advance to $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}}$.
- if $\mathcal{C}(\boldsymbol{\beta}) = \{\boldsymbol{\beta}_1,\ \boldsymbol{\beta}_2\} \subset \mathbb{Z}^n_+$ then $\boldsymbol{x}^{\boldsymbol{\beta}} = \boldsymbol{x}^{\boldsymbol{\beta}_1}\boldsymbol{x}^{\boldsymbol{\beta}_2}$ or $\boldsymbol{\beta} = \boldsymbol{\beta}_1 + \boldsymbol{\beta}_2$; hence the value $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}}$ is computed as the product of the values $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}_1}$ and $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}_2}$.

If either of $\boldsymbol{\beta}_j$ $(j = 1, 2)$ is not a member of $\mathcal{B}$, we add it to $\mathcal{B}$. We also note that if $\mathcal{C}(\boldsymbol{\beta}) = \{\boldsymbol{\beta}_1, \boldsymbol{\beta}_2\}$, then $\kappa(\boldsymbol{\beta}) < \kappa(\boldsymbol{\beta}_1)$ and $\kappa(\boldsymbol{\beta}) < \kappa(\boldsymbol{\beta}_2)$ so that the values $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}_1}$ and $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}_2}$ are computed in advance to the value $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}}$.

Algorithm 5.1.

- Input: A nonempty finite subset $\mathcal{B}$ of $\mathbb{Z}^n_+\backslash\{\mathbf{0}\}$ containing $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n$.
- Output: $\widetilde{\mathcal{B}} \supseteq \mathcal{B}$, $\kappa(\boldsymbol{\beta})$ $(\boldsymbol{\beta} \in \widetilde{\mathcal{B}})$ and $\mathcal{C}(\boldsymbol{\beta}) \in \widetilde{\mathcal{B}} \times \widetilde{\mathcal{B}}$ $(\boldsymbol{\beta} \in \widetilde{\mathcal{B}})$.

Step 1: Let $\ell = 0$ and $\widetilde{\mathcal{B}} = \emptyset$.

Step 2: Let $\delta = \max\{\deg(\boldsymbol{\beta}) : \boldsymbol{\beta} \in \mathcal{B}\}$. If $\delta > 1$ then go to Step 4. Otherwise go to Step 3.

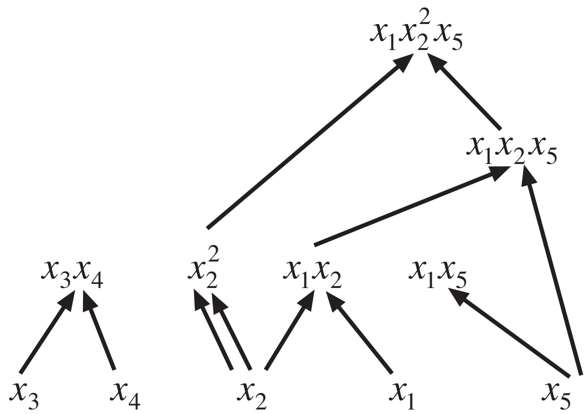

Figure 4: Output of Algorithm 5.1

Step 3: In this case, we have $\mathcal{B} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n\}$. Let $\widetilde{\mathcal{B}} = \widetilde{\mathcal{B}} \cup \mathcal{B}$ and $\mathcal{B} = \emptyset$. Let

$$\kappa(\boldsymbol{e}_i) = \ell + i \ \ (i = 1, 2, \ldots, n), \ \mathcal{C}(\boldsymbol{e}_i) = \emptyset \ (i = 1, 2, \ldots, n).$$

Output $\widetilde{\mathcal{B}} \supseteq \mathcal{B}$, $k(\boldsymbol{\beta})$ $(\boldsymbol{\beta} \in \widetilde{\mathcal{B}})$, and $\mathcal{C}(\boldsymbol{\beta}) \in \widetilde{\mathcal{B}} \times \widetilde{\mathcal{B}}$ $(\boldsymbol{\beta} \in \widetilde{\mathcal{B}})$ and stop.

Step 4: Remove a $\boldsymbol{\beta}$ with $\delta = \deg(\boldsymbol{\beta})$ from $\mathcal{B}$, and let $\widetilde{\mathcal{B}} = \widetilde{\mathcal{B}} \cup \{\boldsymbol{\beta}\}$, $\ell = \ell + 1$ and $\kappa(\boldsymbol{\beta}) = \ell$. If $\boldsymbol{\beta} = \boldsymbol{\beta}_1 + \boldsymbol{\beta}_2$ for some $\boldsymbol{\beta}^1 \in \mathcal{B}$ and $\boldsymbol{\beta}^2 \in \mathcal{B}$, then let $\mathcal{C}(\boldsymbol{\beta}) = \{\boldsymbol{\beta}_1, \ \boldsymbol{\beta}_2\}$ and go to Step 2. Otherwise, go to Step 5.

Step 5: Let

$$\mathcal{C}_1(\boldsymbol{\beta}) = \{\boldsymbol{\beta}' \in \mathcal{B} : \boldsymbol{\beta}' \leq \boldsymbol{\beta}, \ \boldsymbol{\beta}' \neq \boldsymbol{\beta}\},$$

and choose a $\boldsymbol{\beta}_1$ having the largest $\deg(\boldsymbol{\beta}_1)$ from $\mathcal{C}_1(\boldsymbol{\beta})$; $\deg(\boldsymbol{\beta}_1) = \max\{\deg(\boldsymbol{\beta}') : \boldsymbol{\beta}' \in \mathcal{C}_1(\boldsymbol{\beta})\}$. (Note that $\mathcal{C}_1(\boldsymbol{\beta})$ is nonempty since $\boldsymbol{e}_i \in \mathcal{C}_1(\boldsymbol{\beta})$ for every i such that $[\boldsymbol{\beta}]_i \geq 1$. Let $\boldsymbol{\beta}_2 = \boldsymbol{\beta} - \boldsymbol{\beta}_1$, $\mathcal{B} = \mathcal{B} \cup \{\boldsymbol{\beta}_2\}$ and $\mathcal{C}(\boldsymbol{\beta}) = \{\boldsymbol{\beta}_1, \ \boldsymbol{\beta}_2\}$. Go to step 2.

As an example, let $\mathcal{B} = \{x_1x_2^3x_5, x_1x_2x_5, x_1x_2, x_3x_4, x_1x_5, x_i \ (i = 1, 2, \ldots, 5)\}$. Applying Algorithm 5.1, we obtain that

$$\begin{array}{rcl}
\kappa(x_1x_2^3x_5) & = & 1, \ \mathcal{C}(x_1x_2^3x_5) = \{x_1x_2x_5, x_2^2\}, \text{ where } x_2^2 \text{ is added to } \mathcal{B}, \\
\kappa(x_1x_2x_5) & = & 2, \mathcal{C}(x_1x_2x_5) = \{x_1x_2, x_5\}, \\
\kappa(x_1x_2) & = & 3, \ \mathcal{C}(x_1x_2) = \{x_1, x_2\}, \\
\kappa(x_2^2) & = & 4, \ \mathcal{C}(x_2^2) = \{x_2, x_2\}, \\
\kappa(x_1x_5) & = & 5, \ \mathcal{C}(x_1x_5) = \{x_1, x_5\}, \\
\kappa(x_3x_4) & = & 6, \ \mathcal{C}(X_3x_4) = \{x_3, x_4\}, \\
\kappa(x_i) & = & 6 + i, \ \mathcal{C}(x_i) = \emptyset, \\
\widetilde{\mathcal{B}} & = & \mathcal{B} \cup \{x_2^2\}.
\end{array}$$

These output of Algorithm 5.1 can be depicted as Figure 4.

Given a value $\bar{\boldsymbol{x}} \in \mathbb{C}^n$ for the variable vector $\boldsymbol{x}$, the computation of the values $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}}$ for the monomials $\boldsymbol{x}^{\boldsymbol{\beta}} \in \widetilde{\mathcal{B}}$ is carried out by the algorithm below.

Algorithm 5.2.

- Input: A finite set $\widetilde{\mathcal{B}} \subset \mathbb{Z}^n_+ \backslash \{\mathbf{0}\}$ of monomials, $\kappa(\boldsymbol{\beta})$ ($\boldsymbol{\beta} \in \widetilde{\mathcal{B}}$) and $\mathcal{C}(\boldsymbol{\beta})$ ($\boldsymbol{\beta} \in \widetilde{\mathcal{B}}$), which are constructed by Algorithm 5.1, and $\bar{\boldsymbol{x}} \in \mathbb{C}^n$.

- Values of the monomials $\boldsymbol{x}^{\boldsymbol{\beta}} \in \widetilde{\mathcal{B}}$ at $\boldsymbol{x} = \bar{\boldsymbol{x}}$.

Step 1: Let $\ell = \#\mathcal{B}$. Assign the values $\bar{x}_i$ to the monomial $x_i \in \widetilde{\mathcal{B}}$ ($i = 1, 2, \ldots, n$). Let $\ell = \#\mathcal{B} - n$.

Step 2: If $\ell = 0$ then stop.

Step 3: Choose the $\boldsymbol{\beta} \in \widetilde{\mathcal{B}}$ with $\kappa(\boldsymbol{\beta}) = \ell$, and let $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ be the members of $\mathcal{C}(\boldsymbol{\beta})$. Compute $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}}$ as the product of $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}_1}$ and $\bar{\boldsymbol{x}}^{\boldsymbol{\beta}_2}$. Let $\ell = \ell - 1$. Go to Step 2.

The number of multiplications required by the algorithm above amounts to $\#\widetilde{\mathcal{B}} - n$, which is corresponding to the number of monomials with degree greater than one in $\widetilde{\mathcal{B}}$. In the case of Figure 4, we see that one multiplication is required to compute each monomial with degree greater than one; hence the total number of multiplications amounts to $\#\widetilde{\mathcal{B}} - \#\{x_1, x_2, \ldots, x_5\} = 11 - 5 = 6$.

5.3 Total number of multiplications to evaluate a system of polynomials and their partial derivatives

As mentioned in the previous section, the set of monomials which are involved in Honor factorizations of $f_j(\boldsymbol{x})$ ($j = 1, 2, \ldots, m$) and the evaluation of their partial derivatives are computed in advance to the evaluation of the polynomials and their partial derivatives. Taking account of this, we evaluate the total number of multiplications to compute $f_j(\boldsymbol{x})$ ($j = 1, 2, \ldots, m$) and their partial derivatives $\partial f_j(\boldsymbol{x})/\partial x_i$ ($i = 1, 2, \ldots, n, j = 1, 2, \ldots, m$). Suppose that a Hornor factorization of each $f_j(\boldsymbol{x})$ together with a Hornor tree representing the structure of the factorization is obtained ($j = 1, 2, \ldots, m$). To evaluate $f_j(\boldsymbol{x})$, we need to count

(a) every monomial with a positive degree in the leaf node because it is multiplied by some coefficient c_p.

(b) every $\boldsymbol{\gamma}(g, Q)$ generated by Step 4 of Algorithm 2.1 or every monomial attached to an edge of the Hornor tree because it is multiplied to some $\varphi_L(\cdot; g, Q) \in \mathbb{F}[\boldsymbol{x}, f]$.

If we apply the rules (a) and (b) above to the Hornor tree illustrated in Figure 3, we see that

2 multiplications to evaluate $g_3(\bar{\boldsymbol{x}})$,
1 multiplication to evaluate $g_4(\bar{\boldsymbol{x}})$,
2 multiplications to evaluate $g_5(\bar{\boldsymbol{x}})$,
0 multiplication to evaluate $g_6(\bar{\boldsymbol{x}})$,
1 multiplication to evaluate $g_1(\bar{\boldsymbol{x}}) = \bar{x}_2 g_3(\bar{\boldsymbol{x}}) + g_4(\bar{\boldsymbol{x}})$,
1 multiplication to evaluate $g_2(\bar{\boldsymbol{x}}) = \bar{x}_1 \bar{x}_2 \bar{x}_5 g_5(\bar{\boldsymbol{x}}) + g_6(\bar{\boldsymbol{x}})$,
1 multiplication to evaluate $g_0(\bar{\boldsymbol{x}}) = \bar{x}_3 \bar{x}_4 g_1(\bar{\boldsymbol{x}}) + g_2(\bar{\boldsymbol{x}})$.

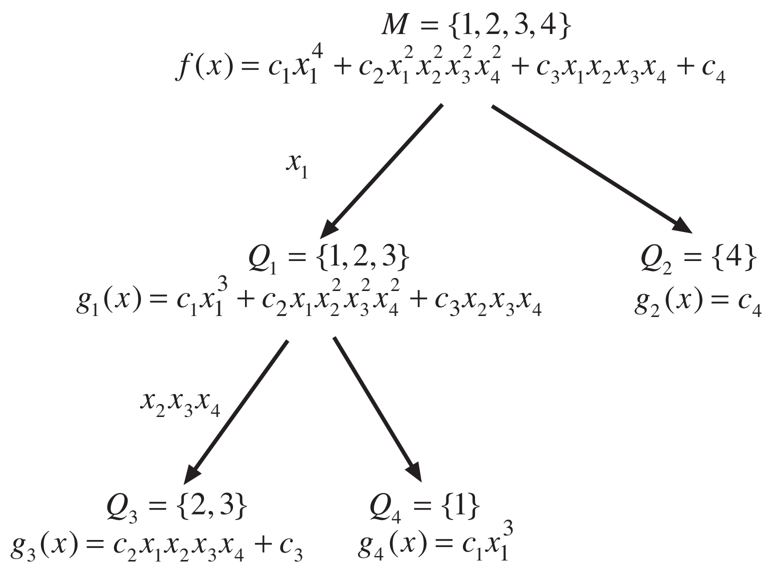

Figure 5: A minimal Hornor factorization of f given in (21)

Thus the total number of multiplications amounts to 8 assuming that the values $\bar{x}_1\bar{x}_2\bar{x}_5$ and $\bar{x}_3\bar{x}_4$ have been computed in advance.

If we recall the chain rule (20) for the partial derivative $\partial g(\boldsymbol{x})/\partial x_i$ of the the polynomial of the form (13), we can apply a similar method as above to count the number of mutiplications required to evaluate $\partial f_j(\boldsymbol{x})/\partial x_i$. Here we only note that the value for $\varphi_L(\boldsymbol{x}; g, Q)$ has been already computed when we evaluate $g(\boldsymbol{x})$, and the details are omitted.

5.4 Example

In this subsection, we show by example that a minimal Hornor factorization of $f \in \mathbb{C}[\boldsymbol{x}]$ dose not necessary result in the minimal number of multiplications when the monomials involved there are efficiently computed in advance. Let us consider a polynomial

$$f(\boldsymbol{x}) = c_1x_1^4 + c_2x_1^2x_2^2x_3^2x_4^2 + c_3x_1x_2x_3x_4 + c_4 \tag{21}$$

for some $c_p \in \mathbb{C}$ ($p \in M = \{1, 2, 3, 4\}$). Then

$$x_1(x_2x_3x_4(c_2x_1x_2x_3x_4 + c_3) + c_1x_1^3) + c_4 \tag{22}$$

is a minimum Hornor factorization, which requires 11 multiplications. See Figure 5.

But if we compute the monomials

$$x_1x_2x_3x_4,\ x_2x_3x_4, x_1^3 \tag{23}$$

in advance, we can reduce the number of multiplications further. Figure 6 illustrates how efficiently we compute these monomials. For example, the first monomial $x_1x_2x_3x_4$ as the product of x_1 and the second monomial $x_2x_3x_4$, the computational of the monomials above requires 5 multiplications. Then, substituting these monomials into the Hornor factorization, we can compute $f(\boldsymbol{x})$ with additional 4 multiplications. Thus we have reduced the number of multiplications from 11 to $9 = 5 + 4$ multiplications.

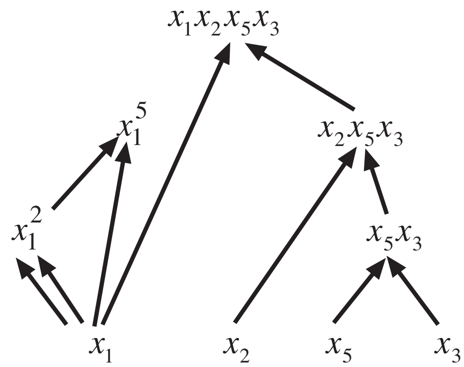

Figure 6: Computation of the monomials involved in (22)

Now we consider a Hornor factorization

$$x_1x_2x_3x_4(c_2x_1x_2x_3x_4 + c_3) + (c_1x_1^4 + c_4), \tag{24}$$

which requires 12 multiplications. In this case, the monomials to be computed in advance are

$$x_1x_2x_3x_4,\ x_1^4.$$

Figure If we compute the first monomial $x_1x_2x_3x_4$ as the product of x_1x_2 and x_3x_4, the number of multiplications for the monomials amounts to 5. Substituting the monomials listed above into the Hornor factorization (24), the computation of $f(\boldsymbol{x})$ is done with additional 3 multiplications. Thus the total number of multiplications amounts to $8 = 5 + 3$, which is smaller than the number of multiplications resulting from the minimum Hornor factorization (22) combined with the efficient computation of the monomials (23).

6 Numerical experiments

We have presented several methods for Hornor factorizations of $f \in \mathbb{C}[\boldsymbol{x}]$. They are:

nu-0: The recursive formula (17). See Section 3.1.

nu-1: The recursive formula (17) with the use of the lower bound λ_1. See Sections 3.2 and 3.3.

nu-2: The recursive formula (17) with the use of the lower bound λ_2. See Sections 3.2 and 3.3.

H1-mu1: Algorithm 4.1 (Heuristic method 1) with the use of $\mu = \mu_1$. See Section 4.1.

H1-mu2: Algorithm 4.1 (Heuristic method 1) with the use of $\mu = \mu_2$. See Section 4.1.

H2: Algorithm 4.2 (Heuristic method 2). See Section 4.2.

H3: Algorithm 4.3 (Heuristic method 3). See Section 4.3.

We report the effectiveness and efficiency of these methods except H1-mu1 through numerical experiments. H1-mu1 was found to be ineffective at all in some preliminary numerical experiments, so that we exclude it here. All the methods were implemented in MATLAB, and the numerical experiments were executed on a Macintosh Dual 2.5GHz powerPC G5 with 2GH DDR SDRAM.

For test problems, 40 systems of polynomials are chosen from Verschelde's web site [17]. Some of their features are shown in Table 1. Each problem is a system of polynomials (with n variables and n equations) of the form (19) with $m = n$. In the table,

$$\left.\begin{array}{rcl} \text{\# of equations} & = & n, \\ \text{max degree} & = & \max\{\deg(f_j) : j = 1, 2, \ldots, n\}, \\ \text{total degree} & = & \displaystyle\sum_{j=1}^{n} \deg(f_j), \\ \text{max \# of terms} & = & \max\{\text{the number of terms of } f_j \ : j = 1, 2, \ldots, n\}, \\ \text{total \# of terms} & = & \displaystyle\sum_{j=1}^{n} \text{the number of terms of } f_j \ . \end{array}\right\} \qquad (25)$$

The values of these features are also attached below the names of the problems in Tables 2, 3 and 4. Among these features, max # of terms is turned out to be the most important feature of a test problem to see whether we can apply the recursive formula (17), which compute the minimum Hornor factorization, to the test problem; as it becomes larger, the number of recursive calls increases rapidly. We classify the test problems in two groups, the one with smaller max # of terms and the other with larger max # of terms. We applied the methods nu-0, nu-1 and nu-2 using the recursive formula (17) only to the former group of problems.

6.1 The recursive formula with and without lower bounds

Table 2 shows the number of multiplications, the number of recursive calls and the cpu time to compute a minimum Hornor factorization when the method nu-0, nu-1 and nu-2 are applied to the group of test problems with smaller max # of terms. Here "# mult" denotes the number of multiplications in a minimum Hornor factorization. We stopped the recursion iteration when the cpu time exceeded 3600 seconds, which is designated by 3600+. From these numerical results, we observe that the method nu-1 combined with the lower bound λ_1 cut the number of recursive calls considerably. The method nu-2 combined with the stronger lower bound λ_2 behaved better than the method nu-1, but the difference is minor in cpu time. This is because the lower bound λ_2 is more expensive than the lower bound λ_1. Practically, these methods are not suitable for large size problems because they would require too much cpu time to compute minimum factorizations of large size problems.

6.2 Comparison of the heuristic methods to the recursive formula (17) with the lower bound λ_2 for small size test problems

Table 3 shows numerical results on the heuristic methods H1-mu2 (Algorithm 4.2 with the use of the upper bound μ_2), H2 (Algorithm 4.2) and H3 (Algorithm 4.3) in comparison to the method nu-2 (the the recursive formula (17) with the lower bound λ_2) when they are

Problem	# of equations	max degree	total degree	max # of terms	total # of terms
chemkin	10	2	17	5	40
game4two	4	3	12	8	32
eco8	8	3	21	8	43
sparse5	5	10	50	8	40
caprasse	4	4	14	9	26
filter9	4	4	16	9	76
butcher	7	4	24	9	55
pb601	3	6	13	9	21
heart	8	4	20	9	48
chemequ	5	3	13	11	30
katsura10	11	2	21	12	107
geneig	6	3	16	15	80
proddeco	4	4	16	15	60
tangents0	6	2	12	16	51
cohn2	4	6	22	16	52
game5two	5	4	20	16	80
cyclic-n $n = 6, 7, 8, 10, 16, 24$	n	n	$\frac{n(n+1)}{2}$	n	$n(n-1)+2$
cohn3	4	6	23	20	74
rose	3	9	19	21	29
sendra	2	7	14	22	26
speer	4	5	20	23	92
cpdm5	5	3	15	23	115
utbikker	4	3	10	27	81
comb3000	4	4	16	29	116
game6two	6	5	30	32	192
rbpl24s	9	3	19	34	103
assur44	8	3	19	41	103
stewgou40	9	4	24	49	199
pole27sys	14	2	28	57	798
game7two	7	6	42	64	448
pole28sys	16	2	32	73	1168
pole34sys	12	3	36	73	876
pole43sys	12	3	36	73	876
rps10	10	4	37	76	688
pltp34sys	12	4	48	96	1152

Table 1: Test problems from Verschelde's web site [17]

		the number of recursive calls (cpu time)		
Problem	# mult	**nu-0**	**nu-1**	**nu-2**
chemkin 10,2,17,**5**,40	47	2 (0.31)	2 (0.10)	2 (0.09)
cyclic6 6,6,21,**6**,32	63	706 (2.41)	535 (1.96)	248 (1.37)
cyclic7 7,7,28,**7**,44	93	7250 (23.46)	5282 (18.64)	2232 (12.12)
game4two 4,3,12,**8**,32	28	228 (0.70)	160 (0.52)	136 (0.48)
eco8 8,3,21,**8**,43	56	551 (1.51)	38 (0.36)	18 (0.32)
cyclic-8 8,8,36,**8**,58	128	84996 (275.79)	61939 (217.61)	24351 (134.15)
sparse5 5,10,50,**8**,40	95	55 (0.34)	30 (0.23)	10 (0.19)
caprasse 4,4,14,**9**,25	40	543 (1.63)	191 (0.83)	125 (0.67)
filter9 4,4,16,**9**,76	88	150 (0.60)	40 (0.36)	34 (0.35)
butcher 7,4,24,**9**,55	70	2211 (5.90)	55 (0.80)	29 (0.74)
pb601 3,6,13,**9**,21	23	956 (2.38)	12 (0.26)	12 (0.26)
heart 8,4,20,**9**,48	100	76 (0.41)	61 (0.36)	49 (0.34)
cyclic-10 10,10,55,**10**,92	-	- (3600+)	- (3600+)	- (3600+)
chemequ 5,3,13,**11**,30	31	442 (1.22)	90 (0.42)	85 (0.41)
katsura10 11,2,21,**12**,107	152	102 (0.58)	78 (0.52)	78 (0.52)
geneig 6,3,16,**15**,80	80	688565 (1793.14)	155108 (634.99)	114068 (551.00)
proddeco 4,4,16,**15**,60	-	- (3600+)	- (3600+)	- (3600+)
tangents0 6,2,12,**16**,51	74	8042 (25.38)	3855 (15.00)	3855 (15.05)
cohn2 4,6,22,**16**,52	-	- (3600+)	- (3600+)	- (3600+)
game5two 5,4,20,**16**,80	-	- (3600+)	- (3600+)	- (3600+)

Table 2: Numerical results on recursive the formula (17) with and without bound

applied to small size test problems. Each box consists of the total number of multiplications in Hornor factorization obtained, the cpu time in seconds, and the total number of multiplications to compute the system of polynomials and their partial derivatives, which is the sum of the numbers of multiplications to compute monomials, function values and derivatives, as presented in Section 5.

First we focus our attention to the number of multiplications in the Hornor factorizations and the cpu time in Table 3. We can confirm that the methods nu-2 attained a Hornor factorization with the smallest number of multiplications among the 4 methods for the problems it was able to process within 3600 seconds. The Hornor factorizations obtained by the less expensive heuristic method H1-mu2 are as good as those obtained by the method nu-2. But both methods become expensive rapidly in cpu time as max # of terms and/or max degree get larger, so that they could be used only for small size problems in practice. On the other hand, the heuristic methods H2 and H3 processed all the problems in Table 3 within 1 seconds. The method H2 attained less numbers of multiplications for some problems including cyclic-6, 7, 8 and 10 than the method H3, but the method H3 behaved better for some other problems including game4two, butcher and geneig.

Concerning the total # of multiplications, we notice that a less number of multiplications of a Hornor factorization does not necessary imply a less total # of multiplications. Recall the example given in Section 5.4. In particular, the minimum Hornor factorization does not necessarily result in the minimum total # of factorizations. See, for example, the cases sparse5, caprasse, filter9, pb601 and cohn2. The method which attained the least value in total # of multiplications varied depending on problems.

6.3 Comparison between the heuristic methods for large size test problems

Table 4 shows numerical results on the heuristic methods H1-mu2, H2 and H3 when they are applied to large size test problems. We notice that the method H1-mu2 was able to process only a few of the test problems within 3600 seconds, so that it could not be used for larger problems in practice. The methods H2 and H3 are much cheaper than the method H1-mu2, and they can be used even for larger problems. None of them behaved better uniformly for all the test problems than the other. Except for cyclic-16 and -24, the total # of multiplications obtained by the method H3 is smaller than or equal to that obtained by the method H3, but, in these two test problems, the quality of the Hornor factorization obtained by the method H2 is much better than that obtained by the method H3. Therefore, we may conclude to use both of them simultaneously in practice; we can choose a better Hornor factorization from the ones generated by them.

7 Concluding discussions

We have proposed a recursive formula for computing a minimum (multivariate) Hornor factorization, a Hornor factorization which requires the minimum number of multiplications to evaluate a multivariate polynomial over all Hornor factorizations. This formula combined with lower bounds for the number of multiplications is effective in computing minimum Hornor factorizations of smaller size polynomials. For larger size polynomials that can not

	# of multiplications in Hornor (cpu time in sec.) the total # of multiplictions (monomials, functions, derivatives)			
Problem	**nu-2**	**H1-mu2**	**H2**	**H3**
chemkin 10,2,17,**5**,40	47 (0.32) 71 (15,32,24)	47 (0.17) 71 (15,32,24)	47 (0.09) 71 (15,32,24)	47 (0.09) 71 (15,32,24)
cyclic6 6,6,21,**6**,32	63 (1.43) 130 (18,46,66)	63 (0.52) 130 (18,46,66)	63 (0.15) 129 (15,46,68)	68 (0.14) 144 (21,46,77)
cyclic7 7,7,28,**7**,44	93 (12.17) 195 (23,65,107)	93 (1.52) 195 (23,65,107)	93 (0.22) 194 (22,65,107)	105 (0.22) 229 (31,65,133)
game4two 4,3,12,**8**,32	28 (0.49) 44 (0,28,14)	28 (0.38) 44 (0,28,14)	32 (0.08) 51 (3,28,20)	28 (0.07) 44 (0,28,14)
eco8 8,3,21,**8**,43	56 (0.32) 125 (11,45,69)	56 (0.71) 125 (11,45,69)	63 (0.13) 130 (19,43,68)	56 (0.11) 125 (11,45,69)
cyclic8 8,8,36,**8**,58	128 (134.65) 279 (29,90,160)	129 (5.38) 277 (32,89,156)	128 (0.30) 279 (29,90,160)	150 (0.31) 342 (44,90,208)
sparse5 5,10,50,**8**,40	95 (0.20) 172 (17,40,115)	95 (0.28) 172 (17,40,115)	100 (0.12) 153 (18,35,100)	110 (0.13) 162 (17,40,105)
caprasse 4,4,14,**9**,25	40 (0.68) 82 (6,28,48)	40 (0.37) 82 (6,28,48)	45 (0.08) 83 (8,25,50)	41 (0.07) 85 (6,29,50)
filter9 4,4,16,**9**,76	88 (0.35) 176 (29,43,104)	89 (0.31) 173 (28,43,102)	91 (0.15) 176 (34,42,100)	89 (0.15) 180 (36,43,101)
butcher 7,4,24,**9**,55	70 (0.74) 127 (21,50,56)	70 (2.13) 127 (21,50,56)	81 (0.17) 138 (24,50,64)	70 (0.15) 127 (21,50,56)
pb601 3,6,13,**9**,21	23 (0.27) 42 (3,21,18)	23 (0.46) 41 (3,20,18)	28 (0.06) 46 (5,20,21)	23 (0.06) 39 (3,19,17)
heart 8,4,20,**9**,48	100 (0.35) 168 (18,52,98)	100 (0.28) 168 (18,52,98)	100 (0.16) 168 (18,52,98)	104 (0.16) 176 (28,52,96)
cyclic-10 10,10,55,**10**,92	- (+3600)	229 (69.69) 519(56,148,315)	228 (0.63) 512(50,148,314)	281 (0.63) 667(77,149,441)
chemequ 5,3,13,**11**,30	31 (0.63) 49 (1,30,18)	31 (0.35) 49 (1,30,18)	34 (0.10) 53 (2,30,20)	31 (0.07) 49 (1,30,18)
katsura10 11,2,21,**12**,107	152 (0.54) 244 (19,130,105)	152 (0.50) 244 (19,130,105)	152 (0.32) 244 (19,130,105)	153 (0.30) 250 (23,129,98)
geneig 6,3,16,**15**,80	- (+3600)	80 (42.97) 130 (1,79,50)	103 (0.25) 159 (6,80,73)	89 (0.14) 144 (0,89,55)
proddeco 4,4,16,**15**,60	- (+3600)	68 (146.31) 140 (0,68,72)	80 (0.21) 159 (3,68,88)	68 (0.16) 140 (0,68,72)
tangents0 6,2,12,**16**,51	74 (15.01) 114 (9,56,49)	74 (0.43) 114 (9,56,49)	74 (0.12) 114 (9,56,49)	74 (0.11) 114 (9,56,49)
cohn2 4,6,22,**16**,52	- (3600+)	- (3600+)	71 (0.18) 138(12,53,73)	62 (0.14) 127 (6,58,63)
game5two 5,4,20,**16**,80	- (3600+)	75 (133.80) 130(0,75,55)	90 (0.26) 151(6,75,70)	75 (0.19) 130(0,75,55)

Table 3: Numerical results on small size test problems

	# of multiplications in Hornor (cpu time in sec.) the total # of multiplictions (monomials, functions, derivatives)		
Problem	**H1-mu2**	**H2**	**H3**
cyclic16 16,.16,136,**16**,242	- (3600+)	718 (2.53) 1702(158,415,1129)	1069 (3.48) 2706(250,421,2035)
cohn3 4,6,23,**20**,74	- (3600+)	103 (0.27) 201(19,76,106)	82 (0.20) 170(7,79,84)
rose 3,9,19,**21**,29	- (3600+)	61 (0.13) 110(15,36,59)	63 (0.10) 97(7,39,51)
sendra 2,7,14,**22**,26	- (3600+)	44 (0.10) 77(8,27,42)	42 (0.08) 74(7,28,39)
speer 4,5,20,**23**,92	- (3600+)	118 (0.31) 225(9,96,120)	92 (0.19) 200(0,92,108)
cpdm5 5,3,15,**23**,115	- (3600+)	156 (0.39) 280(14,115,151)	135 (0.29) 250(5,115,130)
cyclic24 24,24,300,**24**,554	- (3600+)	1923 (11.92) 4770(420,990,3360)	3443 (30.06) 9054(737,1006,7311)
utbikker 4,3,10,**27**,81	77 (508.90) 147(0,77,70)	93 (0.23) 164(9,78,77)	81 (0.47) 145(2,78,65)
comb3000 4,4,16,**29**,116	- (3600+)	164 (0.47) 293(9,124,160)	132 (0.27) 246(2,124,120)
game6two 6,5,30,**32**,192	- (3600+)	234 (0.80) 404(14,186,204)	186 (0.48) 342(0,186,156)
rbpl24s 9,3,19,**34**,103	104 (92.89) 174(10,94,70)	116 (0.34) 192(16,94,82)	104 (0.24) 174(10,94,70)
assur44 8,3,19,**41**,103	- (3600+)	124 (0.35) 207(17,95,95)	104 (0.20) 179(9,95,75)
stewgou40 9,4,24,**49**,199	- (3600+)	255 (0.77) 471(18,195,258)	237 (0.49) 468(18,195,255)
pole27sys 14,2,28,**57**,798	784 (1040.52) 1372(0,784,588)	784 (2.49) 1372(0,784,588)	784 (1.42) 1372(0,784,588)
game7two 7,6,42,**64**,448	- (3600+)	588 (3.46) 1017(30,441,546)	441 (1.60) 840(0,441,399)
pole28sys 16,2,32,**73**,1168	- (3600+)	1152 (3.73) 2048(0,1152,896)	1152 (3.73) 2048(0,1152,896)
pole34sys 12,3,36,**73**,876	- (3600+)	1008 (6.42) 1740(12,864,864)	864 (3.11) 1584(0,864,720)
pole43sys 12,3,36,**73**,876	- (3600+)	1008 (6.37) 1740(12,864,864)	864 (3.10) 1584(0,864,720)
rps10 10,4,37,**76**,638	- (3600+)	984 (6.00) 1656(24,714,918)	777 (2.53) 1534(1,768,765)
pltp34sys 12,4,48,**96**,1152	- (3600+)	1560 (9.61) 2949 (21,1296,1632)	1212 (3.12) 2580 (0,1212,1368)

Table 4: Numerical results on medium and large size test problems

be handled by the recursive formula, we have proposed heuristic methods for computing a Hornor factorization with a less number of multiplications.

Founded on these Hornor factorizations, we have discussed how efficiently we evaluate a system of polynomials and their partial derivatives in homotopy continuation methods, and reported numerical results on 40 test problems. The recursive formula combined with a lower bound for multiplications can effectively process small size test problems in a little cpu time, but it becomes too expensive rapidly as the size of a problem to process gets larger. The proposed heuristic methods H2 and H3 can process such a large problem.

As far as the author knows, there has been little literature on the subject of this paper, how efficiently we evaluate a system of polynomials and their partial derivatives in homotopy continuation methods. This paper is just a beginning of the subject, and there remain many issues to study further. Our numerical experiments are not enough to judge whether the proposed heuristics H2 and H3 work effectively and efficiently in practice. More numerical experiments and better heuristics may be necessary. Also we have not paid any attention to round off errors which would occur in evaluating polynomials and their partial derivatives. This is also an important factor that should be taken into account when we design heuristic methods for evaluating polynomials and their partial derivatives.

References

[1] E. Allgower and K. Georg, *Numerical continuation methods,* Springer-Verlag, 1990.

[2] D. N. Bernshtein, "The number of roots of a system of equations," *Functional Analysis and Appl.* **9** (1975) 183–185.

[3] F. J. Drexler, "Eine methode zur Berechnung sämtlicher Lösungen von Polynomgleichungssystemen," *Numer. Math.* **29** (1977) 45–58

[4] C. B. Garcia and W. I. Zangwill, "Determining all solutions to certain systems of nonlinear equations," *Mathematics of Operations Research* **4** (1979) 1–14.

[5] T. Gao, T. Y. Li, and X. Li, HOM4PS, http://www.mth.msu.edu/ li/, Dept. of Mathematics, Michigan State university, East Lansing, MI 48824, 2005.

[6] T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa and T. Mizutani, "PHoM – a Polyhedral homotopy continuation method for polynomial systems," *Computing.* **73** (2004) 55–77.

[7] B. Huber and B. Sturmfels, "A Polyhedral method for solving sparse polynomial systems," *Mathematics of Computation* **64** (1995) 1541–1555.

[8] T. Y. Li, "Solving polynomial systems," *The mathematical intelligencer* **9** (1987) 33–39.

[9] T. Y. Li, "Solving polynomial systems by polyhedral homotopies", *Taiwan Journal of Mathematics* **3** (1999) 251–279.

[10] A. Morgan, *"Solving polynomial systems using continuation for engineering and scientific problems,"* Prentice-Hall, 1987.

[11] A. P. Morgan and A. J. Sommese, *"Coefficient-parameter polynomial continuation," Appl. Math. Comput.* **29** (1989) 123–160.

[12] J. M. Pena and T. Sauer, "On the multivariate Hornor scheme," *SIAM J. Numer. Anal.* **37** (2000) 1186-1197.

[13] J. M. Pena and T. Sauer, "On the multivariate Hornor scheme II: Running error analysis," *Computing* **65** (2000) 313-322.

[14] L. B. Rall and G. F. Corliss, "An introduction to automatic differentiation", L. B. Rall, ed. *Automatic Differentiation — Techniques and Applications*, Lecture Notes in Computer Science, Vol. 120, Springer-Verlag (1981).

[15] B, Sturmfels, *Solving systems of polynomial equations*, CBMS Regional Conference Series in Mathematics, No 97, American Mathematical Society, 2002.

[16] H. J. Su , J. M. McCarthy , M. Sosonkina and L. T. Watson , "POLSYS_GLP: A Parallel General Linear Product Homotopy Code for Solving Polynomial Systems of Equations", To appear in the Algorithms section of *ACM Trans. Math. Softw.*,

[17] J. Verschelde, The database of polynomial systems is in his web site: "http://www.math.uic.edu/∼jan/".

[18] J. Verschelde, P. Verlinden and R. Cools, "Homotopies exploiting Newton polytopes for solving sparse polynomial systems," *SIAM J. Numerical Analysis* **31** (1994) 915–930.

[19] J. Verschelde, "Homotopy continuation methods for solving polynomial systems," *Ph.D. thesis, Department of Computer Science, Katholieke Universiteit Leuven,* 1996.

[20] J. Verschelde, "Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation," *ACM Trans. Math. Softw.* **25** (1999) 251–276.

[21] L. T. Watson, M. Sosonkina, R. C. Melville, A. P. Morgan, and H. F. Walker, "HOMPACK90: A suite of Fortran 90 codes for globally homotopy algorithms," *ACM Trans. Math. Softw.* **23** (1997) 514–549.

[22] J. H. Wilkinson, *Rounding Errors in Algebraic Processes,* Notes Appl. Sci. 32, her Majesty's Stationery Office, London, 1963.