# Construction of Janet Bases
# I. Monomial Bases

Vladimir P. Gerdt[1], Yuri A. Blinkov[2], Denis A. Yanovich[1]

[1] Laboratory of Information Technologies
Joint Institute for Nuclear Research, 141980 Dubna, Russia
gerdt@jinr.ru yan@cv.jinr.ru
[2] Department of Mathematics and Mechanics
Saratov University, 410071 Saratov, Russia
BlinkovUA@info.sgu.ru

**Abstract.** Algorithms for computation of Janet bases for monomial ideals and implementation of these algorithms are presented. As data structures for finite monomial sets the binary trees called Janet trees are selected. An algorithm for construction of a Janet basis for the ideal generated by a finite monomial set is described. This algorithm contains as subalgorithms those to search for Janet divisor in a given tree and to insert monomials into the tree in the process of completion to involution. The algorithms presented have been implemented in C in the form of package for completion of monomial sets to Janet involutive ones. An example is given to illustrate practical efficiency of the monomial algorithms and their implementation.

## 1 Introduction

In papers [1–3] we invented a concept of involutive division, designed general algorithms for construction of polynomial involutive bases and studied a number of particular divisions. In paper [4] these algorithms were generalized to linear systems of partial differential equations. For Pommaret division the polynomial involutive algorithms were implemented in Reduce [1, 5]. The algorithms for construction of monomial involutive bases and the computation of the Hilbert functions and Hilbert polynomials based on them were implemented in Mathematica [6]. The latter code designed for the general involutive division and written in the Mathematica language is much slower than the straightforward implementation of Janet division in C as we demonstrated in [6]. The Mathematica implementation has been recently extended to compute involutive polynomial and linear differential bases[1].

In the given paper we present algorithms and data structures to compute monomial Janet bases with particular features of Janet division taken into account. By this reason these algorithms are more efficient then those designed for the general involutive division and applied to construction of Janet bases.

[1] The package is available at http://paul.math-inf.uni-greifswald.de/Invo/.

The paper is organized as follows. Section 2 contains the necessary definitions and notations to be used in the sequel.

In Section 3 we describe the data structure for a finite monomial set as a binary tree called Janet tree. The elements in the set are associated with the leaves of the tree. This data structure is amenable to efficient search for Janet divisor and for completion of monomial sets to Janet bases.

An algorithm of the fast search for Janet divisor in a given tree is presented in Section 4. For this algorithm we give an estimation of its complexity in comparison to that of the classical binary search algorithm.

In Section 5 we describe algorithms to insert into a given Janet tree of a monomial which has no involutive divisors among the leaf monomials in the tree and for computation of nonmultiplicative prolongations which arise from the insertion.

The main algorithm for computation of a minimal monomial Janet basis is described in Section 6. This algorithm is a specification to Janet division of the general algorithm which was put forward in [3] for the general involutive division. The given specification contains as subalgorithms all those ones described in Sections 3-4 and unlike the general algorithm there is no need in the conventional autoreduction at the initialization step.

Section 7 illustrates practical efficiency of the proposed algorithms by means of their implementation in C. One of important applications of monomial involutive bases is their convenience for computation of the Hilbert function and Hilbert polynomial [4, 7]. In this context the efficiency of monomial computations is illustrated by example from paper [8].

## 2 Janet Division

In this paper we use the following notations: $\mathbb{N}$ is the set of non-negative integers; $\mathbb{M} = \{x_1^{d_1} \cdots x_n^{d_n} \mid d_i \in \mathbb{N}\}$ is the set of monomials in the polynomial ring $\mathbb{R} = \mathbb{K}[x_1, \ldots, x_n]$ over zero characteristic field $\mathbb{K}$; $\deg_i(u)$ is the degree of $x_i$ in $u \in \mathbb{M}$; $\deg(u) = \sum_{i=1}^{m} \deg_i(u)$ is the total degree of $u$; $\succ$ is an admissible [9–11] monomial ordering such that

$$x_1 \succ x_2 \succ \cdots \succ x_n \,. \tag{1}$$

If monomial $u$ divides monomial $v$ we shall write $u \mid v$.

An involutive monomial division $L$ [1] for any finite monomial set $U \subset \mathbb{M}$ and any its element $u \in U$ assigns the partition $\{x_1, \ldots, x_n\} = M_L(u, U) \cup NM_L(u, U)$ of variables into multiplicative $x_i \in M_L(u, U)$ and nonmultiplicative $x_i \in NM_L(u, U)$. Respectively, $u \in U$ is an $L-$(involutive) divisor of $w \in \mathbb{M}$, if $u \mid w$ and monomial $w/u$ contains only multiplicative variables for $u$.

A number of involutive divisions were discovered and studied to date [1–3, 7, 12]. In the present paper we consider one of the divisions which corresponds to the classical separation of variables into multiplicative and nonmultiplicative invented by Janet for the purpose of algebraic analysis of partial differential equations [13].

**Definition 1.** Janet division [1]. Let $U \subset \mathbb{M}$ be a finite monomial set. For each $1 \leq i \leq n$ divide $U$ into groups labeled by non-negative integers $d_1, \ldots, d_i$:

$$[d_1, \ldots, d_i] = \{ \ u \ \in U \mid d_j = \deg_j(u), \ 1 \leq j \leq i \ \}. \tag{2}$$

A variable $x_i$ is called (Janet) *multiplicative* for $u \in U$ if $i = 1$ and

$$\deg_1(u) = \max\{\deg_1(v) \mid v \in U\},$$

or if $i > 1$, $u \in [d_1, \ldots, d_{i-1}]$ and $\deg_i(u) = \max\{\deg_i(v) \mid v \in [d_1, \ldots, d_{i-1}]\}$. If a variable is not $J-$multiplicative for $u \in U$, it is called (Janet) *nonmultiplicative* for $u$. In the latter case we shall write $x_i \in NM_J(u, U)$. If $u \in U$ is a Janet divisor of monomial $w$ we shall write $u \mid_J w$.

**Definition 2.** [1] A finite monomial set $U$ is called Janet involutive if

$$(\forall u \in U) \ (\forall x \in NM_J(u, U)) \ (\exists v \in U) \ \ [ \ v \mid_J (u \cdot x) \ ] \, . \tag{3}$$

To prove correctness of the main algorithm described in Section 7 we need the following lemma.

**Lemma 1.** *Let $U \subset \mathbb{M}$ be a finite set and the set $\tilde{U} \supseteq U$ be Janet involutive. Then $\tilde{U}$ contains any nonmultiplicative prolongation $u \cdot x$ of an element $u \in U$ ($x \in NM_J(u, U)$) such that $u \cdot x$ has no Janet divisor in $U$ and among the conventional divisors of $u \cdot x$ there are no other nonmultiplicative prolongations of elements in $U$ also without Janet divisors in $U$.*

*Proof.* The given property of Janet division follows from its constructivity defined in paper [1] where in the proof of Theorem 4.14 it is shown that the prolongation under consideration of the above lemma belongs to the involutive completion of the initial set.

**Definition 3.** [2] A Janet involutive set $U$ is called a minimal Janet basis of the ideal $Id(U) \subset \mathbb{R}$, generated by $U$ if for any other Janet basis $V$ of this ideal the inclusion $U \subseteq V$ holds. This Janet basis is unique [2].

## 3 Data Structures

By definition of Janet division, a monomial set $U$ is partitioned into groups (2) indexed by degrees of the first $i$ variables. The variable $x_{i+1}$ is multiplicative for $u \in U$ if $\deg_{i+1}(u)$ takes the maximal value among monomials in the group, and nonmultiplicative, otherwise. For the variable $x_1$ the whole monomial set $U$ must be considered as a group.
Below we construct a binary tree which we shall call Janet tree. Its structure reflects the partition of elements in $U$ into the groups (2) which are sorted in the degrees of variables within every group. We demonstrate that such a data

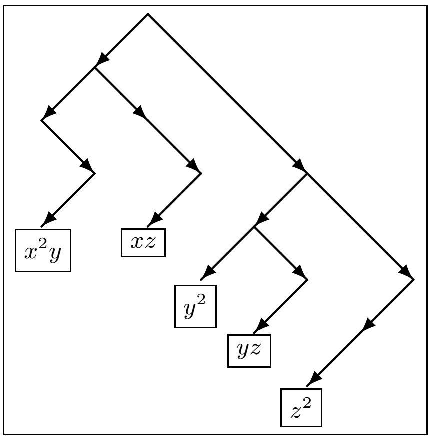
structure allows one to search efficiently for a Janet divisor and also to compute all the multiplicative variables for elements in $U$.

Before proceeding to the description of the general structure of Janet trees we explain it in terms of the following specific example.

*Example 1.* $U = \{x^2y, xz, y^2, yz, z^2\}$ $(x \succ y \succ z)$.

| *Monomial* | *Variables* | |
|---|---|---|
| | *nonmultiplicative* | *multiplicative* |
| $x^2y$ | $-$ | $x, y, z$ |
| $xz$ | $x$ | $y, z$ |
| $y^2$ | $x$ | $y, z$ |
| $yz$ | $x, y$ | $z$ |
| $z^2$ | $x, y$ | $z$ |

Portray the monomial set of Example 1 in the form of Janet tree as indicated in the following picture. As shown in this picture, monomials of the given set $U$ are




assigned to the leaves of the tree. The monomial with increased by one degree of the current variable is assigned to the left child whereas the right child points at the next variable with respect to chosen ordering.

Now consider the structure of Janet tree of the general form as a set $JT := \cup\{\nu\}$ of internal nodes and leaves which corresponds to the nonempty monomial set. To every element $\nu$ of the tree we shall assign the set of three elements $\nu = \{u,\, nd,\, nv\}$ with the following structure:

$$\begin{array}{lll} \mathrm{mon}(\nu) & = u & \text{is the monomial assigned to } \nu, \\ \mathrm{ndg}(\nu) & = nd & \text{is the pointer to the next node in degree,} \\ \mathrm{nvr}(\nu) & = nv & \text{is the pointer to the next node in variable.} \end{array}$$

In the absence of a child we shall assign the value *nil* to the corresponding pointer. Wherever it does not lead to misunderstanding we shall identify the pointers $nd$ and $nv$ with the nodes they point out. If the subtree $jt \subset JT$ of the Janet tree is rooted at $\nu \in JT$ we shall write $jt = \mathrm{jst}(\nu)$ and $\nu = root(jt)$.

The unit monomial is assigned to the root of a Janet tree $JT$ which will be denoted by $\mathrm{root}(JT)$. The internal nodes and leaves of tree $JT$ are characterized by the following states:

$$\text{Internal node: } (nv \neq nil) \vee ((nd \neq nil) \wedge (\deg(\mathrm{mon}(nd)) = deg(u) + 1))$$
$$\text{Leaf: } (nv = nil) \wedge (nd = nil)$$

## 4 Search for Janet Divisor

Given a Janet tree $JT$ and a monomial $w$, the following algorithm provides a fast search for Janet divisor of $w$ among the monomials assigned to leaves of $JT$.

**Algorithm: J-divisor($JT$, $w$)**

**Input:** $JT$, a Janet tree; $w$, a monomial
**Output:** the leaf monomial $u \mid_J w$ in $JT$ or $nil$, otherwise
1: $\nu := \mathrm{root}(JT)$
2: **while** $\mathrm{mon}(\nu) \mid w$ **do**
3:   **while** $\mathrm{ndg}(\nu)$ **and** $\mathrm{mon}(\mathrm{ndg}(\nu)) \mid w$ **do**
4:     $\nu := \mathrm{ndg}(\nu)$
5:   **od**
6:   **if** $\mathrm{nvr}(\nu)$ **then**
7:     $\nu := \mathrm{nvr}(\nu)$
8:   **elif** $\mathrm{ndg}(\nu)$ **then**
9:     **return** *nil*
10:   **else**
11:     **return** $\mathrm{mon}(\nu)$
12:   **fi**
13: **od**

Prove correctness of this algorithm. By Definition 1, to be a Janet divisor of $w$, the monomial $u \in U$ must be the lexicographically highest either among all the monomials in $U$ or among monomials in the group (2) with $d_1 = \deg_1(w), \ldots, d_i = \deg_i(w)$ and some $1 \leq i < n$. As this takes place $\deg_j(w) \geq \deg_j(u) + 1$ for $j > i$.
Therefore, the correctness of algorithm **J-divisor** follows from the fact that after termination of the inner **while**-loop 3–5 there may be three alternate situations.

1. $(\mathrm{ndg}(\nu) = nil) \wedge (\mathrm{nvr}(\nu) \neq nil)$. In this case one needs to continue the search for divisor by transition to the next, by order (1), variable that is done in line 7.

2. $ndg(\nu) \neq nil$. If this holds, then there is no Janet divisor. Indeed, the sequential transition to the left in line 7 would end up with the leaf whose monomial has a higher degree in the current variable than $w$. Apparently, such a monomial cannot divide $w$ both conventionally and involutively.
3. $(\mathrm{ndg}(\nu) = nil) \wedge (\mathrm{nvr}(\nu) = nil)$. This means that $\nu$ is a leaf of the tree and the monomial assigned to this leaf if just the desired Janet divisor.

The next theorem shows that complexity bound of the proposed algorithm **J-divisor** is lower than that of the classical binary search algorithm for Janet divisor in the lexicographically sorted array of monomials has been implemented earlier [6].

**Theorem 1.** *Let $d$ be the maximal total degree of monomials in $n$ variables which constitute the finite set $U$. Then the complexity bound of the algorithm* $\mathbf{J-divisor}$ *and the binary search algorithm is given by*

$$\begin{aligned} t_{\mathbf{J-divisor}(U)} &= O(d+n), \\ t_{\mathbf{BinarySearch}(U)} &= O(n((d+n)\log(d+n) - n\log(n) - d\log(d))). \end{aligned}$$

*Proof.* The complexity of the algorithm $\mathbf{J-divisor}$ is determined by the total length of walk in the related Janet tree. Since the total degree of an assigned monomial is bounded by $d$, the total length of walk in the tree is obviously bounded by $d+n$.

Furthermore, the structure of the Janet tree is such that when the transition to the next monomial holds in the process of the Janet divisor search this new monomial is different from the foregoing one by degree in a single variable only. This allows one to verify the Janet divisibility only with respect to that variable. As a result the computing time of the search is restricted by $t_{\mathbf{J-divisor}(G)} = O(d+n)$ instead of $O\left((d+n)\,n\right)$.

The number of distinct monomials in $n$ variables and of the total degree $\leq d$ is $\binom{d+n}{n}$. Complexity of the binary search in a sorted list or array of $k$ elements is $O(\log_2 k)$. Let $k$ be a number of elements in $U$. Then $k \leq \binom{d+n}{n}$. To estimate the expression

$$O\left(\log_2 \binom{d+n}{n}\right) = O\left(\log\left(\frac{(d+n)!}{n!d!}\right)\right)$$

we use the Stirling formula in the form

$$\log(n!) - ((n+1/2)\log(n) - n) = O(1)\,.$$

Then, taking into account the fact that the comparison of monomials which is done in the course of the binary search algorithm has complexity $O(n)$, we obtain:

$$t_{\mathbf{BinarySearch}(U)} = O\left(n\left((d+n)\log(d+n) - n\log(n) - d\log(d)\right)\right).$$

## 5 Insertion of Monomials into Janet Tree

Describe now an algorithm **OneleafTree**$(u,\, i,\, w)$ which for $w \mid u$ constructs the rooted binary tree with a single leaf. The constructed tree is rooted at monomial $w$, and its leaf monomial is $w \prod_{j=i}^{n} x_i^{\deg_j(u)-\deg_j(w)}$.

The internal nodes of the constructed tree receive the structure similar to the structure of Janet trees described in Section 3. In so doing the inner **while**-loop successively generates the left descendants of the current node with growing degrees of their monomials in the current variable $x_i$ $(1 \le i \le n)$. Generation of the right children is done in line 13 of the outer **while**-loop 3–16. Correctness of this algorithm is obvious.

**Algorithm: OneleafTree**$(u,\, i,\, w)$

**Input:** $u$, a monomial; $i$, the index of a variable $x_i$; $w$, a divisor of $u$
**Output:** *tree*, the one-leaf tree rooted at $w$ and having $w \prod_{k=i}^{n} x_k^{deg_k(u)}$ in the leaf

1: $\nu := \{w,\, nil,\, nil\}$
2: $tree := \{\nu\}$
3: **while** $u \neq w$ **and** $i \le n$ **do**
4:   **while** $w \cdot x_i \mid u$ **do**
5:     $w := w \cdot x_i$
6:     $\mathrm{ndg}(\nu) := \{w,\, nil,\, nil\}$
7:     $\nu := \mathrm{ndg}(\nu)$
8:     $tree := tree \cup \{\nu\}$
9:   **od**
10:   **if** $u \neq w$ **then**
11:     $\mathrm{nvr}(\nu) := \{w,\, nil,\, nil\}$
12:     $\nu := \mathrm{nvr}(\nu)$
13:     $tree := tree \cup \{\nu\}$
14:     $i := i + 1$
15:   **fi**
16: **od**
17: **return** *tree*

The following algorithm **J-prolong**$(jt, i, V)$ is recursive. It is applied to the subtree of a Janet tree such that variable $x_i$ is nonmultiplicative for all the leaf monomials in the subtree. As part of the algorithm all nonmultiplicative prolongations of these monomials in $x_i$ are determined and added to the set $V$ which is used in the below described main algorithm **MonomialJanetBasis**.

It might be well to point out that the structure of the algorithm **J-prolong** is inherent in the bypassing algorithms for Janet trees. This structure shows that all the leaves of the input subtree $jt$ are reached as part of the algorithms. Therefore, its correctness follows from the fact that at every leaf reaching the

**Algorithm: J-prolong($jt$, $i$, $V$)**

**Input:** $jt$, a Janet subtree; $i$, the index of variable $x_i$ which is nonmultiplicative for all the leaf monomials in $jt$; $V$, a monomial set
**Output:** the set $V$ enlarged with nonmultiplicative prolongations in $x_i$ of the leaf monomials in $jt$

1: $\nu := \mathrm{root}(jt)$
2: **while** $\mathrm{ndg}(\nu)$ **do**
3:   **if** $\mathrm{nvr}(\nu)$ **then**
4:     *J-prolong*($\mathrm{jst}(\nu)$, $i$, $V$)
5:   **fi**
6:   $\nu := \mathrm{ndg}(\nu)$
7: **od**
8: **if** $\mathrm{nvr}(\nu)$ **then**
9:   *J-prolong*($\mathrm{jst}(\nu)$, $i$, $V$)
10: **else**
11:   $V := V \cup \{\mathrm{mon}(\nu) \cdot x_i\}$
12: **fi**
13: **return** $V$

nonmultiplicative prolongation of the corresponding monomial in $x_i$ is computed in line 11 and added to the monomial set $V$.

The algorithms **OneleafTree** and **J-prolong** enter as subalgorithms in the algorithm **J-insert** which inserts a monomial in a given Janet tree.

As the algorithm **J-insert** works, besides adding on the new subtrees to the Janet tree due to the insertion of monomial $u$, all the new nonmultiplicative prolongations induced by the insertion are also computed. These prolongations are added to the set $V$. It should be noted that the lack of Janet divisors of $u$ among the leaf monomials in the tree is essentially exploited when $u$ is inserted. This is indicated explicitly in the input data of the algorithm.

To show correctness of the algorithm **J-insert** consider two alternatives which may occur on completion in the **while**-loop 5-7:

1. $(\deg_i(\mathrm{mon}(\nu)) < \deg_i(u)) \wedge (\mathrm{ndg}(\nu) = nil)$. In this case the inserted monomial $u$ has a higher degree in $x_i$ than monomial in the current node of the tree. In so doing the equality $\deg_j(\mathrm{mon}(\nu)) = \deg_j(u)$ holds for all $j < i$ if $i > 1$. This is ensured by the assignment in line 13 which provides quitting the loop and transition to line 29 after execution of lines 9-12. Thereby, the insertion leads to conversion of $x_i$ from multiplicative to nonmultiplicative for all the leaf monomials in the subtree rooted at that node. Respectively, in line 9 computation of all the new nonmultiplicative prolongations for the current branch is done. In lines 10–12 the tree is added on with the one-leaf tree having $u$ in the leaf.
2. $\deg_i(\mathrm{mon}(\nu)) = \deg_i(u)$. If $\mathrm{ndg}(\nu) \neq nil$, then $x_i$ is a nonmultiplicative variable for monomial $u$ inserted in the tree, and the corresponding prolongation

**Algorithm: J-insert**$(JT,\ u,\ V)$

**Input:** $JT$, a Janet tree; $u$, a monomial to be inserted in $J$ and such that *J-divisor*$(JT,\ u)$ returns *nil*; $V$, a monomial set

**Output:** $JTree$, the Janet tree obtained from $JT$ by inserting $u$; the set $V$, enlarged with nonmultiplicative prolongations of the leaf monomials in $JTree$ arising from insertion of $u$

```
i := 1
JTree := JT
ν := root(JTree)
while i ≤ n do
   while deg_i (mon(ν)) < deg_i(u) and ndg(ν) do
      ν := ndg(ν)
   od
   if deg_i(mon(ν)) < deg_i(u) then
      J-prolong(jst(ν), i, V)
      jtree := OneleafTree(u, i, mon(ν) · x_i)
      ndg(ν) := root(jtree)
      JTree := JTree ∪ jtree
      i := n + 1
   else
      if ndg(ν) then
         V := V ∪ {u · x_i}
      fi
      if nvr(ν) then
         i := i + 1
         ν := nvr(ν)
      else
         if i < n then
            jtree := OneleafTree(u, i + 1, mon(ν))
            ndg(ν) := root(jtree)
            JTree := JTree ∪ jtree
            i := n + 1
         fi
      fi
   fi
od
return JTree, V
```

is added to $V$ in line 16. Furthermore, if $\mathrm{nvr}(\nu) \neq nil$ the outer **while**-loop is being continued by transition to the next variable. Otherwise one needs only add on the one-leaf tree as done in lines 23-25 and quit the outer loops afterwards that is provided by assignment in line 25.

Therefore, the correctness of algorithm **J-insert** follows from correctness of its subalgorithms **OneleafTree** and **J-prolong** has been established.

Apparently, all the considered algorithms are terminated that governs termination of the main algorithm **MonomialJanetBasis** described in the next section.

## 6 Main Algorithm

Given a finite monomial set $U$ the below algorithm is a specific form of the general involutive algorithm [3] for construction of the minimal Janet basis of ideal $Id(U)$.

The initial Janet tree defined in line 1 of the algorithm has the root only. As prescribed in Section 3, the unit monomial is assigned to the root. If the input monomial set contains the unit monomial that is verified in lines 3-4, then the algorithm outputs $\bar{U} = \{1\}$. Otherwise, in the **while**-loop the set $\bar{U}$ initialized in line 6 as an empty set is augmented by the elements in the set $V$ without Janet divisors in the current Janet tree $JT$.

Due to line 12, as explained in the previous section, the set $V$, in addition to elements in $U$ which have not been chosen in line 8 yet, contains nonmultiplicative prolongations of elements in $\bar{U}$. These prolongations according to Definition 2 must be checked for existence of their Janet divisors in $\bar{U}$. Such a check is done in line 10, and if the nonmultiplicative prolongation under consideration has no Janet divisor it is inserted in $\bar{U}$ in line 11. Similarly, the elements of the initial monomial set $U$ are proceeded in lines 10-11 in addition to the nonmultiplicative prolongations.

Let us now prove the correctness of the algorithm **MonomialJanetBasis**. By the condition in line 8, all those monomials in $U$ which have no conventional divisors among elements in $\bar{U}$ are to be inserted in $\bar{U}$ since every Janet divisor is also the conventional divisor [1]. Therefore, the output set $\bar{U}$ is a basis of the monomial ideal $Id(U)$.

We must prove that this output basis satisfies the conditions (3) of Janet involutivity and is minimal. Denote by $U_0$ the minimal Janet basis of $Id(U)$ and by $W$ the set $\bar{U}$ in some intermediate stage of the algorithm. Assume that $W \subset U_0$. This obviously holds after insertion of the first element in the initially empty set $\bar{U}$.

Let set $V$ contain the nonmultiplicative prolongation $u \cdot x_i \in V$ of an element $u \in W$ such that there are no conventional divisors of this prolongation in $V$ except the prolongation $u \cdot x_i$ itself. Then, if this prolongation is free from Janet divisor in $W$, by Lemma 1, the inclusion $u \cdot x_i \in U_0$ holds, and, hence, $W \cup \{u \cdot x_i\} \subseteq U_0$ in compliance with insertion done in line 12.

**Algorithm: MonomialJanetBasis($U$)**

**Input:** $U$, a finite set of distinct monomials
**Output:** $\bar{U}$, the minimal Janet basis of ideal $Id(U)$
1: $JT := \{\{1, nil, nil\}\}$
2: $V := U$
3: **if** $1 \in V$ **then**
4: $\quad\bar{U} := \{1\}$
5: **else**
6: $\quad\bar{U} := \emptyset$
7: $\quad$**while** $V \neq \emptyset$ **do**
8: $\quad\quad$**choose** $u \in V$ without divisor in $V \setminus \{u\}$
9: $\quad\quad V := V \setminus \{u\}$
10: $\quad\quad$**if** *J-divisor*$(JT, u) = nil$ **then**
11: $\quad\quad\quad\bar{U} := \bar{U} \cup \{u\}$
12: $\quad\quad\quad$*J-insert*$(JT, u, V)$
13: $\quad\quad$**fi**
14: $\quad$**od**
15: **fi**
16: **return** $\bar{U}$

Suppose now that $V$ contains an element $u \in U$ such that $u$ is free from the conventional divisors in $V \setminus \{u\}$ but there is such a divisor in $W$. If $u$ has several such divisors, let $v$ be the highest of them with respect to the lexicographical ordering induced by (1). Then from the lack of Janet divisors in $W$ for $u$ it follows that $u = v \cdot w$, where $v \in W$ and $w$ contains variables nonmultiplicative for $v$. Let $x_k \in NM_J(v, W)$ be the highest (with the least subscript $k$) among such variables. By Definition 1 of Janet division, there is a monomial $u_1 \in W$ in the group containing $v$ and such that $\deg_k(v) < \deg_k(u_1)$. But then the nonmultiplicative prolongation $v \cdot x_k$ must be examined in line 8 before monomial $u$. As this takes place, $v \cdot x_k \notin W$. Otherwise, since $v \cdot x_k \mid u$, the monomial $v$ could not be lexicographically highest among divisors of $u$. On the other hand, if $W$ would contain a Janet divisor $v_1$ of $v \cdot x_k$, then $v_1$ would be lexicographically higher than $v$ and would satisfy $v_1 \mid u$, a contradiction. It follows that $u$ has a Janet divisor in $W$.

Thus, we have shown that the initial conventional autoreduction of the input monomial set needs not be performed in the given algorithm as distinct from the general involutive algorithm described in [3].

Therefore, the correctness of the main algorithm **MonomialJanetBasis** reduces to correctness of its subalgorithms **J-divisor** and **J-insert** which has been proved. Termination of the main algorithm for arbitrary finite monomial set $U$ follows from termination of the indicated subalgorithms and noetherity of the Janet division [1].

## 7 Implementation in C

The above described algorithms and data structures have been implemented in C. In the same manner as in the previous C implementation [6] the presentation of monomial $x_1^{i_1} \cdots x_n^{i_n}$ by the array $[i_1, \ldots, i_n]$ of its exponents is used. However, in contrast to the previous implementation where the classical binary search algorithm for Janet divisor was used, the given implementation which rests on the above algorithm **J-divisor** is far more efficient. This is in agreement with theoretical comparison of both search algorithms discussed in Section 4. Experimental comparison of the previous and new implementations shows that the latter is several times faster.

Besides, as well as in the previous implementation the Pommaret division [1] was also implemented. This division also admits data structures in the form of binary trees which are slightly different from Janet trees. We implemented the Pommaret division for the purpose of experimental comparison of both divisions in computation of involutive bases. Our experiment has shown that computation of monomial Pommaret bases does not reveal notable distinction in speed in comparison with Janet bases. In doing so, a minimal Janet basis coincides with a Pommaret basis whenever the latter is finite [14]. Though Janet division, unlike Pommaret division, calls for extra computation to recalculate the separation of variables into multiplicative and nonmultiplicative caused by insertion of a new monomial in the binary tree, for Janet division one often needs examine less number of prolongations than for Pommaret division. This is because the algorithm **MonomialJanetBasis** enables the fulfillment of conditions stated in Proposition 3.10 of paper [1] which imply that for any monomial in the intermediate basis its Janet nonmultiplicative variable is also Pommaret nonmultiplicative. As this takes place, the converse is not generally true.

In order to provide non-existence of conventional divisors for the choice of element in $V$ in line 8 of the algorithm **MonomialJanetBasis** it suffices to sort $V$ with respect to any admissible monomial ordering and choose the lowest monomial. In the given implementation the following three admissible orderings which are widely used in computation of Gröbner bases [10, 11] were built-in:

- lexicographical,
- degree lexicographical,
- degree reverse lexicographical.

In the monomial case, in contrast to the polynomial case, the choice of admissible ordering for selection of nonmultiplicative prolongations is not of crucial importance. This is because not only the output Janet basis which is uniquely defined by the input monomial set [2] but also the number of steps which are performed in the **while**-loop 7-14, are invariant under the choice of ordering [3]. The speed of computation may nevertheless depend on choice of ordering, and for Janet division the lexicographical ordering is most optimal [3]. The other built-in orderings are important in polynomial computations, for which the present implementation is an intermediate stage.

To illustrate practical efficiency of the proposed algorithms and their implementation we consider the following example, taken from paper [8] where it is used for computation of the Hilbert series implicitly representing the Hilbert function [10, 11].

*Example 2. Let $U_m$ ($m \in \mathbb{N}$) be the leading monomial set for the reduced Gröbner basis of the polynomial ideal generated by the elements of the squared $m \times m$ matrix $A$*

$$A = \left\| \begin{matrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,m} \end{matrix} \right\| ,$$

*whose $m^2$ elements are considered as variables. The Gröbner basis is computed for the degree reverse lexicographical ordering extending the following ordering on the variables*

$$x_{1,1} \succ x_{1,2} \succ \cdots \succ x_{1,m} \succ x_{2,1} \succ \cdots \succ x_{2,m} \succ \cdots \succ x_{m,1} \succ \cdots \succ x_{m,m} \, .$$

*The below table gives the quantitative characteristics of the input monomial set $U_m$ for $m = 4, 5$, the output Janet basis $\bar{U}_m$ and the computing times on a Pentium III 700 Mhz based PC running under RedHat Linux 6.2. In the last column of the table the dimension of the monomial ideal $Id(U_m)$ is given.*

| *m* | *Number of variables* | *Size of* $U_m$ | *Size of* $\bar{U}_m$ | *Computing time (sec)* | *Dimension of ideal* |
|---|---|---|---|---|---|
| *4* | *16* | *161* | *1324* | *0.03* | *8* |
| *5* | *25* | *1372* | *72302* | *14.55* | *12* |

The dimension of the ideals listed in the table can be determined as the degree of the Hilbert polynomial $HP(s)$, which as well as the Hilbert function $HF(s)$ are easily computed in terms of the monomial Janet basis $\bar{U}$ by the explicit formulae [4, 7]

$$HP(s) = \binom{n+s}{s} - \sum_{u \in \bar{U}} \binom{s - \deg(u) + \mu(u)}{\mu(u)} ,$$

$$HF(s) = \binom{n+s}{s} - \sum_{i=0}^{s} \sum_{u \in \bar{U}} \binom{i - \deg(u) + \mu(u) - 1}{\mu(u) - 1} .$$

Here $n$ is the number of variables and $\mu(u)$ is the number of multiplicative variables for monomial $u$.

For the monomial ideals generated by $U_4$ and $U_5$ in Example 2, the Hilbert polynomials $HP_4(s)$ and $HP_5(s)$ are given by

$$HP_4(s) = \frac{1}{1440}\,s^8 + \frac{1}{72}\,s^7 + \frac{8}{45}\,s^6 + \frac{113}{144}\,s^5 + \frac{3259}{1440}\,s^4 + \frac{617}{144}\,s^3 + \frac{607}{120}\,s^2 + \frac{41}{12}\,s + 3$$

$$HP_5(s) = \frac{1}{1088640}\,s^{12} + \frac{1}{30240}\,s^{11} + \frac{121}{217728}\,s^{10} + \frac{209}{36288}\,s^9 + \frac{3223}{72576}\,s^8 + \frac{3173}{15120}\,s^7 + \frac{174367}{217728}\,s^6 + \frac{134699}{60480}\,s^5 + \frac{2480627}{544320}\,s^4 + \frac{152101}{22680}\,s^3 + \frac{19949}{3024}\,s^2 + \frac{1213}{315}\,s + 3$$

## 8 Conclusion

The procedures implementing slightly modified versions of algorithms described in Sections 4 and 5 form a part of our Reduce, C and C++ packages for computation of polynomial Janet bases. Instead of algorithm **MonomialJanetBasis** in the polynomial case the algorithm described in [3] is used with some extra optimizations. This improved algorithm together with its implementation and the timings for numerous benchmarks widely used by designers of the Gröbner bases software is presented in our second paper devoted to computation of polynomial Janet bases [15].

As we noticed in Sect.7 the binary tree data structure is also inherent in Pommaret division. We do not know to which extent such a structure is also inherent in other involutive divisions constructed in [1–3, 12]. But the recent results obtained in [16] outline a way to construction such a tree for a general involutive division.

This work was partially supported by the RFBR grants 00-15-96691, 01-01-00708 and by grant INTAS 99-1222.

## References

1. Gerdt, V. P., Blinkov, Yu. A.: Involutive Bases of Polynomial Ideals. *Math. Comp. Simul.* **45** (1998) 519–542
2. Gerdt, V. P., Blinkov, Yu. A.: Minimal Involutive Bases. *Math. Comp. Simul.* **45** (1998) 543–560
3. Gerdt, V. P.: Involutive Division Technique: Some Generalizations and Optimizations, *Zapiski Nauchnykh Seminarov POMI (St.Petersburg)* **258** (1999) 185–206. To be published in *J. Math. Sci.*
4. Gerdt, V. P.: Completion of Linear Differential Systems to Involution. In: *Computer Algebra in Scientific Computing / CASC'99*, V.G. Ganzha, E.W. Mayr and E.V. Vorozhtsov (Eds.), Springer-Verlag, Berlin (1999) 115–137
5. Zharkov, A. Yu., Blinkov, Yu. A.: Involutive Approach to Investigating Polynomial Systems. *Math. Comp. Simul.* **42** (1996) 323–332

6. Gerdt, V. P., Kornyak, V.V., Berth, M., Czichowsky G.: Construction of Involutive Monomial Sets for Different Involutive Divisions. In: *Computer Algebra in Scientific Computing / CASC'99*, V. G. Ganzha, E.W. Mayr and E.V. Vorozhtsov (Eds.), Springer-Verlag, Berlin (1999) 147–157
7. Apel, J.: Theory of Involutive Divisions and an Application to Hilbert Function. *J. Symb. Comp.* **25** (1998) 683–704
8. Bayer, D., Stillman, M.: Computation of Hilbert Functions. *J. Symb. Comp.* **14** (1992) 31–50
9. Buchberger, B.: Gröbner Bases: an Algorithmic Method in Polynomial Ideal Theory. In: *Recent Trends in Multidimensional System Theory*, N.K. Bose (ed.), Reidel, Dordrecht (1985) 184–232
10. Becker, T., Weispfenning, V., Kredel, H.: *Gröbner Bases. A Computational Approach to Commutative Algebra*. Graduate Texts in Mathematics **141**, Springer-Verlag, New York (1993)
11. Cox, D., Little, J., O'Shea, D.: *Ideals, Varieties and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra.* 2nd Edition, Springer-Verlag, New-York (1996)
12. Chen Yu-Fu, Gao Xiao-Shan.: Vector Representation of Involutive Divisions. *Mathematics-Mechanization Research Preprints* **No. 18**, Beijing (1999) 9–22
13. Janet, M.: *Leçons sur les Systèmes d'Equations aux Dérivées Partielles*, Cahiers Scientifiques, IV, Gauthier-Villars, Paris (1929)
14. Gerdt, V. P.: On the Relation Between Pommaret and Janet Bases. In: *Computer Algebra in Scientific Computing/ CASC 2000*, V.G. Ganzha, E.W. Mayr, E.V. Vorozhtsov (Eds.), Springer-Verlag, Berlin (2000) 167–181
15. Gerdt, V. P., Blinkov, Yu. A., Yanovich, D. A.: Construction of Janet Bases. II. Polynomial Bases. This volume.
16. Blinkov, Yu. A.: The Method of Separative Monomials for Involutive Divisions. *Programming and Computer Software* **3** (2001) 43–45