J. Symbolic Computation (1996) **11**, 1–000

KANT V4

M. Daberkow, C. Fieker, J. Klüners, M. Pohst, K. Roegner, M. Schörnig, and K. Wildanger:

Technische Universität Berlin,

Fachbereich 3, Sekr. MA 8-1,

Straße des 17. Juni 136, D–10623 Berlin, Germany

E-mail address:daberkow, fieker, klueners, pohst, roegner, schoern, wildan@math.tu-berlin.de

(Received 19 April 1996)

The software package KANT V4 for computations in algebraic number fields is now available in version 4. In addition a new user interface has been released. We will outline the features of this new software package.

1. Introduction

KANT V4 is a program package for computations in algebraic number fields. The emphasis is on the interaction of elements of subfields of a given field. Consequently, algebraic integers are considered to be elements of a specified order of an appropriate number field $\mathcal{F}$. Algebraic numbers are presented by an integer and a denominator, usually chosen as a natural number. The available algorithms provide the user with the means to compute all invariants of $\mathcal{F}$ and to solve tasks like calculating the solutions of Diophantine equations related to $\mathcal{F}$. Further subfields of $\mathcal{F}$ can be generated and $\mathcal{F}$ can be embedded into an overfield. The potential of moving elements between different fields (orders) is a significant feature of our system.

KANT V4 was developed at the University of Düsseldorf from 1987 until 1993 and at the Technical University Berlin afterwards. During these years the performance of existing algorithms and their implementations grew dramatically. While calculations in number fields of degree 4 and up were nearly impossible before 1970 and number fields of degree more than 10 were beyond reach until 1990, it is now possible to compute in number fields of degree well over 20, and – in special cases – even beyond 1000. This also characterizes one of the principles of KANT V4, namely to support computations in number fields of

0747–7171/90/000000 + 00 $03.00/0

arbitrary degree rather than fixing the degree and pushing the size of the discriminant to the limit.

Our philosophy is to substitute mere existence theorems for invariants of algebraic number fields by algorithms which actually provide these objects. In the seventies H. Zassenhaus postulated the following 4 principal tasks for computational algebraic number theory:

> the development of efficient algorithms for the computation of
>
> > the maximal order,
> > the unit group,
> > the class group,
> > the Galois group
>
> of an algebraic number field $\mathcal{F}$.

All these tasks were solved effectively in the past, though we are far from having efficient methods in general. However, those algorithms are fundamental for recent research in advanced topics like the computation of class fields (see below). In the meantime various new algorithms were established, and all the old ones were improved, some of them considerably. Members of the KANT group have contributed considerably to this progress. Hence, bottlenecks of existing implementations could be overcome by improvements of the theory rather than by efforts to write better code.

KANT V4 consists of a C–library of more than 1000 functions for doing arithmetic in number fields. Of course, the necessary auxiliaries from linear algebra over rings, especially lattices, are also included. The set of these functions is based on the computer algebra system MAGMA [Cannon (1996)] from which we adopt our storage management, arithmetic for (long) integers and arbitrary precision floating point numbers, arithmetic for finite fields, polynomial arithmetic and a variety of other tools. Essentially, all of the public domain part of MAGMA is contained in KANT V4 . In return, almost all KANT V4 routines are included in MAGMA. The only other computer algebra system allowing extensive calculations with algebraic numbers and number fields is PARI (Bordeaux). Systems like LiDIA and SIMATH (both Saarbrücken) offer only access to a rather limited number of tasks.

A user of KANT V4 routines needs to write his own header programs, which requires some knowledge of the storage handling in MAGMA. To make KANT V4 easier to use we recently developed a shell called KASH . This shell is based on that of the group theory package GAP [Schönert et al. (1993)] and the handling is similar to that of MAPLE. We put great effort into enabling the user to handle the number theoretical objects in the very same way as one would do using pencil and paper. For example, there is just one command **Factor** for the factorization of elements from a factorial monoid like rational integers in $\mathbb{Z}$, polynomials over a field, or ideals from a Dedekind ring.

In the subsequent sections we discuss in some detail:

the realization of number theoretical objects in KANT V4 and the corresponding data types;
the most important algorithms contained in KANT V4;
the shell KASH in greater detail to demonstrate its power and simple use;
the potential of KANT V4 as well as KASH to carry out tasks on several workstations simultaneously, for distributed calculations we make use of PVM;
the integrated SQL–database for number fields.

The article is concluded by several introductory and also more sophisticated examples.

The development of KANT V4 as well as KASH is continued in view of providing the user with the most advanced tools for computations in algebraic number fields. In the near future we plan to install

additional (basic) routines into KASH,
a general machinery for local fields,
a general machinery for global fields,
a Diophantine equation solver for special classes of equations.

Suggestions for additional important features to be included are welcome.

2. Basic Concepts and Data Structures

The design of KANT V4 is based on the mathematical structures of algebraic number fields. We have recently extended this design to function fields over finite fields, for which the following structures of algebraic number fields will soon be available.

The design of the structures has been laid out to minimize limitations, which are due to practical aspects. As a result nearly all mathematical objects in KANT V4 can have several repesentations and some even at the same time. Among the available objects, the following are the most important:

orders,
algebraic numbers,
ideals in orders,
lattices.

The most important data type is the order type. For simplicity we define $\mathbb{Z}$ to be the trivial order and $\mathbb{Q}$ to be the trivial number field. An order $\mathcal{O}$ is a data type, which collects information about arithmetic for elements in the quotient field $\mathcal{F} = Q(\mathcal{O})$ of $\mathcal{O}$ having a representation related to the order (see below) and information concerning the order itself, such as the discriminant, unit group or regulator. For the maximal order $o_{\mathcal{F}}$, some general information about $\mathcal{F}$, such as the class number or the class group, can be stored.

2.1. Orders

In KANT V4 an order $\mathcal{O}_1$ is always a free module with another order $\mathcal{O}_2$ as its coefficient ring. The degree of $\mathcal{O}_1$ over $\mathcal{O}_2$ equals the field degree $[Q(\mathcal{O}_1) : Q(\mathcal{O}_2)]$ of the associated quotient fields. There are basically two ways to create orders. Both methods assume that we already have an order $\mathcal{O}$ (remember that $\mathbb{Z}$ is treated as an order). The first method can be used to create an order in a finite extension of the quotient field of the given order $\mathcal{O}$. This is done by a monic polynomial $f(t) \in \mathcal{O}[t]$ which is irreducible in $Q(\mathcal{O})[t]$. The generated order is $\mathcal{O}[\alpha]$, with the $\mathcal{O}$–basis $1, \alpha, \dots, \alpha^{\deg(f)-1}$ if α is an algebraic number with $f(\alpha) = 0$. An order created in this way is called the equation order of α over $\mathcal{O}$. The second method can only be applied to non trivial orders $\mathcal{O}$. In this case $\mathcal{O}$ is an $\mathcal{O}'$ module with

$$\mathcal{O} = \mathcal{O}'\omega_1 + \cdots + \mathcal{O}'\omega_n.$$

An overorder M of $\mathcal{O}$ in $Q(\mathcal{O})$ can now be defined by a transformation $(d, T) \in \mathbb{Z}^{\geq 1} \times \mathcal{O}'^{n \times n}$ such that $\eta_1, \dots, \eta_n$ with

$$(\eta_1, \dots, \eta_n) = \frac{1}{d}(\omega_1, \dots, \omega_n)T$$

is an $\mathcal{O}'$–basis of M.

2.2. Algebraic numbers

To define algebraic numbers in KANT V4 it is necessary to have a non trivial order $\mathcal{O} = \mathcal{O}'\omega_1 + \cdots + \mathcal{O}'\omega_n$. KANT V4 supports two different presentations for an algebraic number $\alpha \in Q(\mathcal{O})$. The first one is the basis presentation of α with respect to the basis of the order $\mathcal{O}$. Here α can be uniquely represented by coefficients $(d, (\alpha_1, \dots, \alpha_n)) \in \mathbb{Z}^{\geq 1} \times \mathcal{O}'^n$ such that

$$\alpha = \frac{1}{d}\left(\alpha_1\omega_1 + \cdots + \alpha_n\omega_n\right).$$

The second method is only available if $\mathcal{O}$ is a $\mathbb{Z}$–order, e.g. if the coefficient ring $\mathcal{O}'$ equals $\mathbb{Z}$. In this case KANT V4 can identify α by a conjugate vector $\mathrm{con}(\alpha)$ as outlined below.

Since $\mathcal{O}$ is a $\mathbb{Z}$–order, we know that a suitable suborder M of $\mathcal{O}$ is an equation order created by a root ρ of a monic, irreducible polynomial $f(t) \in \mathbb{Z}[t]$, i.e. $M = \mathbb{Z}[\rho]$. The second representation of algebraic numbers depends on an appropriate ordering of the roots of f. We denote by r_1 the number of real roots and by $2r_2$ the number of complex roots of f and hence have

$$n = \deg(f) = r_1 + 2r_2.$$

The roots of f are ordered such that $\rho = \rho^{(1)}, \dots, \rho^{(r_1)}$ are the real roots and $\rho^{(r_1+1)}, \dots, \rho^{(r_1+2r_2)}$ are the complex roots of f with $\overline{\rho^{(r_1+j)}} = \rho^{(r_1+r_2+j)}$ $(1 \leq j \leq r_2)$.

Using these definitions, we can define n different, so called conjugate maps on $\mathcal{F} = Q(M) = \{\sum_{i=0}^{n-1} \alpha_i \rho^i \mid \alpha_i \in \mathbb{Q}\}$ by

$$\varphi_k : Q(M) \to \mathbb{C} : \sum_{i=0}^{n-1} \alpha_i \rho^i \mapsto \sum_{i=0}^{n-1} \alpha_i \left(\rho^{(k)}\right)^i \qquad (1 \le k \le n).$$

We call $\varphi_k(\mathcal{F}) =: \mathcal{F}^{(k)}$ the k–th conjugate field of $\mathcal{F}$ and define the conjugate representation $\mathrm{con}(\alpha)$ of an algebraic number $\alpha \in Q(\mathcal{O})$ by an n–tuple $(\alpha_1, \dots, \alpha_n) \in \mathbb{R}^n$ such that

$$\mathrm{con}(\alpha) = (\varphi_1(\alpha), \dots, \varphi_{r_1}(\alpha), \Re(\varphi_{r_1+1}(\alpha)), \Im(\varphi_{r_1+1}(\alpha)), \dots, \Re(\varphi_{r_1+r_2}(\alpha)), \Im(\varphi_{r_1+r_2}(\alpha)))^{\mathrm{tr}}.$$

It is possible to perform arithmetic with algebraic numbers in each of the available representations.

2.3. Ideals

The fractional ideals of a $\mathbb{Z}$–order $\mathcal{O} = \mathbb{Z}\omega_1 + \cdots + \mathbb{Z}\omega_n$ are another important structure within KANT V4. As we know, these ideals form a group for the ring of integers $o_\mathcal{E}$ of $\mathcal{E} = Q(\mathcal{O})$. Since the algorithmic treatment of relative extensions is very new, we will not consider ideals of orders in relative extensions here, but rather restrict ourselves to the case of absolute orders with a $\mathbb{Z}$–basis. KANT V4 supports two different formats for these ideals. One is available in all orders and the other is restricted to maximal orders. The general representation is based on the fact that every fractional ideal $\mathfrak{a}$ is a free $\mathbb{Z}$–module, i.e. there are $\alpha_1, \dots, \alpha_n \in \mathfrak{a}$ with

$$\mathfrak{a} = \mathbb{Z}\alpha_1 + \cdots + \mathbb{Z}\alpha_n.$$

Thus we are able to represent the fractional ideal $\mathfrak{a}$ of $\mathcal{O}$ by a tuple $(d, M) \in \mathbb{Z}^{>0} \times \mathbb{Z}^{n\times n}$ such that

$$(\alpha_1, \dots, \alpha_n) = \frac{1}{d}(\omega_1, \dots, \omega_n)M.$$

For doing computations with this ideal we use special representations, e.g. for testing whether two ideals are equal. Here the Hermite normal form plays an important role. For example, that representation is unique if M is in Hermite normal form (column reduced) and no prime number p which divides d does divide all entries of M. We note that an ideal in KANT V4 can have several different $\mathbb{Z}$–bases at the same time.

The second representation is basically only available for the ideals of the maximal order, so that we assume $\mathcal{O}$ is a maximal order $o_\mathcal{E}$ of some algebraic number field $\mathcal{E}$. This representation is the so called 2–element representation and is based on the fact that for every fractional ideal $\mathfrak{a}$ of $o_\mathcal{E}$ we can find two elements $\alpha, \beta \in \mathfrak{a}$ such that

$$\mathfrak{a} = \alpha o_\mathcal{E} + \beta o_\mathcal{E}.$$

A slightly refined version of this 2–element representation is the so called P–normal representation, which is a 2–element representation of $\mathfrak{a} = \alpha o_\mathcal{E} + \beta o_\mathcal{E}$ such that for a given set of prime ideals $P = \{\mathfrak{p}_1, \dots, \mathfrak{p}_k\}$ the following conditions are satisfied

$$\alpha \in o_\mathcal{E} \ \text{ and } \ \forall\, \mathfrak{p} \in P\ : \alpha\mathfrak{a}^{-1} \not\subseteq \mathfrak{p}.$$

This somehow artificial representation provides fast arithmetic for ideals $\mathfrak{a} = \alpha o_{\mathcal{E}} + \beta o_{\mathcal{E}}$ and $\mathfrak{b} = \gamma o_{\mathcal{E}} + \delta o_{\mathcal{E}}$. We have, for example, $\mathfrak{a} \cdot \mathfrak{b} = \alpha\gamma o_{\mathcal{E}} + \beta\delta o_{\mathcal{E}}$ for P–normal representations and a fast inversion of ideals is possible in a similar way [Pohst, Zassenhaus (1989)].

Beside these two different representations, KANT V4 can store several other pieces of information related to an ideal. Some examples are the norm of the ideal, a single generator if the ideal is principal, or the ramification index and the degree of inertia if the ideal is a prime ideal.

2.4. Lattices

The final important number theoretical structure we want to discuss is the lattice structure, which arises in a natural way from Minkowski's Geometry of Numbers. The important lattices here are those which are defined by ideals and orders having a $\mathbb{Z}$–basis. Given an ideal or order $M = \mathbb{Z}\,\omega_1 + \cdots + \mathbb{Z}\,\omega_n$ to define the lattice L associated to M we need the map $\mathrm{mink}(\alpha) : M \to \mathbb{R}^n$ defined by

$$\begin{aligned} \mathrm{mink}(\alpha) = (\varphi_1(\alpha), \ldots, \varphi_{r_1}(\alpha), \sqrt{2}\Re(\varphi_{r_1+1}(\alpha)), \sqrt{2}\Im(\varphi_{r_1+1}(\alpha)), \\ \ldots, \sqrt{2}\Re(\varphi_{r_1+r_2}(\alpha)), \sqrt{2}\Im(\varphi_{r_1+r_2}(\alpha)))^{\mathrm{tr}}, \end{aligned}$$

which is a normalized version of the con–map. L is then given by the set

$$L(M) := \{x = (x_1, \ldots, x_n)^{\mathrm{tr}} \in \mathbb{R}^n \mid x = \mathrm{mink}(\alpha) \text{ for some } \alpha \in M\}.$$

The importance of lattices for constructive number theory cannot be stressed often enough. They are an important structure for performing computations in number fields, since they offer access to algebraic numbers in a canonical way. Besides the definition of a lattice via ideals and orders, KANT V4 also offers the possibility to define arbitrary lattices by a Gram–matrix, i.e. a positive definite matrix $G \in \mathbb{R}^{n\times n}$ for some $n \in \mathbb{Z}^{\geq 1}$. At the moment the data which can be stored in a lattice range from the successive minima of the lattice and its discriminant to the Gram–matrix and the quadratic form associated to the lattice.

3. Library functions and algorithms

In this section we give a short overview of the KANT V4 library and the most important algorithms. Many of the library functions can be accessed by using the shell. This will be discussed in a later section.

3.1. Basics

The C–library of KANT V4 is based on the MAGMA library, which is under development by J. Cannon in Sydney. This library offers a memory management package, an arbitrary precision integer and real number package, and a generic treatment of matrices and polynomials.

3.2. Lattices

In the sequel we will consider some of the algorithms provided by the KANT V4 library which can be applied on a given lattice $\Lambda = \mathbb{Z}\,\mathbf{b}_1 + \cdots + \mathbb{Z}\,\mathbf{b}_k \subset \mathbb{R}^n$ of rank k with $\mathbb{R}$-linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k$.

The first task considered is the computation of a basis of *short* lattice vectors [Pohst, Zassenhaus (1989)]. Ordering the vectors of $\mathbb{R}^n$ according to their norm induces a partial ordering $\prec$ on the set B_Λ of all bases of Λ. A minimal element of B_Λ with respect to $\prec$ is called a Minkowski reduced basis. However, the computation of Minkowski reduced bases can be rather time consuming. Therefore in KANT V4 , LLL-reduced bases are computed as a substitute. Although their properties are not as nice as the ones of Minkowksi reduced bases, LLL-reduced bases usually suffice for most applications.

In some cases however, it is necessary to compute all vectors $\mathbf{b}$ of Λ subject to $\|\mathbf{b}\|_2 \leq C$ for a certain constant $C > 0$. In KANT V4, all these vectors are determined by the enumeration algorithm of Fincke/Pohst. The KANT V4 library also provides slight modifications of the enumeration algorithm for the computation of a shortest non-zero lattice vector and for the determination of all lattice vectors which are close to a given vector $\mathbf{v}$ in the subspace of $\mathbb{R}^n$ generated by Λ.

Additionally KANT V4 offers functions for determining a basis from a system of generators by the MLLL algorithm and for the computation of the successive minima of a lattice.

3.3. Fundamental functions for number fields

For a given number field $\mathcal{F}$, represented by an order $\mathcal{O}$, KANT V4 offers the complete arithmetic of elements in $\mathcal{F}$. For ideals of arbitrary $\mathbb{Z}$ orders addition and multiplication are possible. Ideals of maximal orders can be inverted and factorized into prime ideals; this includes the decomposition law of rational primes in $\mathcal{F}$. For arbitrary orders this is possible only, if the ideal under consideration is coprime to the index of that order. All elementary operations for polynomials and matrices are adopted from MAGMA.

According to H. Zassenhaus, one of the founders of computational algebraic number theory, the main tasks in this area are the computation of an integral basis, the unit group, the class group and the Galois group.

Integral Bases An integral basis for the maximal order $o_\mathcal{F}$ of a number field $\mathcal{F}$ is computed by a combination of the Round–2 and Round–4 algorithm [Cohen (1993), Pohst (1993), Pohst, Zassenhaus (1989)]. Let $f(t) \in \mathbb{Z}[t]$ be a monic irreducible polynomial such that $\mathcal{F}$ is generated by a root ρ of $f(t)$. For each prime number p whose square divides the discriminant $d(f)$ of f we determine its p–maximal overorder R_p defined by

$$R_p = \{\alpha \in o_\mathcal{F} \mid \exists\, m \in \mathbb{N} : p^m \alpha \in \mathbb{Z}[\rho]\}.$$

Because of

$$o_{\mathcal{F}}/\mathbb{Z}[\rho] = \bigoplus_{p^2 | d(f)} R_p/\mathbb{Z}[\rho],$$

we easily obtain the maximal order $o_{\mathcal{F}}$ from all p–maximal orders. We note that the Dedekind criterion allows us to decide whether the equation order $\mathbb{Z}[\rho]$ is already p–maximal for a given prime p. The Round–4 method is essentially used to split the task of determining p–maximal orders into several such tasks, but for polynomials of smaller degrees.

Fundamental Units For the computation of the unit group in the maximal order $o_{\mathcal{F}}$ we use different strategies. A generalized continued fraction algorithm [Pohst (1993)] computes such a system of independent units $\varepsilon_1, \ldots, \varepsilon_r, r = r_1 + r_2 - 1$, if the field discriminant is "small". We note that for any given set $I \subset \{1, \ldots, r_1+r_2\}$ this algorithm calculates a unit ε subject to

$$\begin{array}{rcll} |\varphi_i(\varepsilon)| & < & 1 & \forall\, i \in I, \\ |\varphi_j(\varepsilon)| & \geq & 1 & \forall\, j \in \{1, \ldots, r_1 + r_2\} \setminus I. \end{array}$$

However, for larger number fields we obtain a system of independent units by the relation method, we calculate integers of $\mathcal{F}$ generating principal ideals which factor over a given set of prime ideals [Cohen (1993)]. Set

$$G \;= < \zeta > \times < \varepsilon_1 > \times \cdots \times < \varepsilon_r >,$$

where ζ is a generator of the torsion units in $o_{\mathcal{F}}$. After computing a lower regulator bound, we obtain an upper bound B for the index of G in the full unit group of $o_{\mathcal{F}}$. By testing

$$\omega^p \in G \quad (\omega \in o_{\mathcal{F}} \setminus G, p \in \mathbb{P}, p \leq B)$$

for solvability we extend G to the full unit group. Unit groups in arbitrary orders R are obtained by intersecting the unit group of the maximal order with R.

Class Groups The computation of the class group $Cl_F = I_F/H_F$ is done by a combination of the methods from Pohst, Zassenhaus (1989) and Cohen (1993).

We first assume that the class group is generated by classes of prime ideals $\mathfrak{p}_1, \ldots, \mathfrak{p}_s$ with norm below some given bound. This bound is usually chosen quite small relative to the Minkowski bound for which the above assumption is known to be true. The correctness of the small bound can be proved in an extra step.

Next we approximate the class group from above by finding relations between the classes $\mathfrak{p}_1 H_F, \ldots, \mathfrak{p}_s H_F$, i. e. algebraic numbers α such that

$$\alpha o_F = \prod_{i=1}^{s} \mathfrak{p}_i^{k_i}.$$

These relations can be obtained in different ways. The enumeration of short vectors in lattices of conjugates associated to the $\mathfrak{p}_i$ usually yields relations in a fast manner. We also apply ideal reduction or a strategy to force independent relations. The idea of considering exceptional primes from the Number Field Sieve essentially supports the relation search in difficult cases.

In this way we obtain an overgroup of the class group. We then determine the class group itself by taking p-th roots similar to the procedure for the computation of fundamental units. However, assuming GRH, it is often better to use an approximation to the Euler product which satisfies

$$\prod_p \frac{(1-p^{-1})}{\prod_{\mathfrak{p}\,|\,p}(1-N(\mathfrak{p})^{-1})} = \frac{2^{r_1}(2\pi)^{r_2} h_F R_F}{w\sqrt{|d_F|}}$$

to check if we are done and to continue with the relation search otherwise. Afterwards, we can apply a verification procedure to make the results independent from GRH. These computations give the class group via integral ideals $\mathfrak{a}_1, \ldots, \mathfrak{a}_k$, algebraic elements $\alpha_1, \ldots, \alpha_k$ and positive minimal integers $c_1, \ldots, c_k$ with $c_i \mid c_{i+1}$ such that $\mathfrak{a}_i^{c_i} = \alpha_i o_F$ and

$$Cl_F \;= <\mathfrak{a}_1\, H_F> \times \cdots \times <\mathfrak{a}_k\, H_F> \simeq\; (\mathbb{Z}/c_1\mathbb{Z}) \times \cdots \times (\mathbb{Z}/c_k\mathbb{Z}).$$

Galois Groups Let $f(t) \in \mathbb{Z}[t]$ denote an irreducible polynomial of degree n. Its Galois group Γ is a transitive subgroup of S_n, the symmetric group of n letters. For an approximation of Γ from below, the modulo p factorizations of f for a few primes p give a rather precise idea of the cycle types to be expected in Γ, excluding all transitive subgroups of S_n which are *too small*. Using factorization methods over number fields, the inductive construction of $\mathbb{Q}(\rho_1, \ldots, \rho_s)$ for some roots $\rho_1, \ldots, \rho_s$ of f will usually produce an approximation of Γ from above. In the case that both approximations still admit several candidates for Γ we need to decide whether Γ is contained in one of certain conjugate subgroups of S_n. This decision is possible through the use of indicator functions. Presently, KANT V4 enables the computation of Galois groups for polynomials up to degree 11 [Pohst, Zassenhaus (1989), Eichenlaub, Olivier (1996)].

Although the results of KANT V4 concerning unit groups and class groups do *not* generally depend on the validity of GRH, by using certain options, the user has the possibility of assuming GRH to speed up the computation of fundamental units and of class groups.

3.4. Sophisticated features

The fundamental functions for number fields enabled the implementation of more sophisticated and specialized features. We list those which are currently realized. Implementations were partly done using the programming language of the shell.

Subfields Let $\mathcal{F} = \mathbb{Q}(\alpha)$ be an algebraic number field which is given by a zero α of the corresponding minimal polynomial $f \in \mathbb{Z}[t]$. Each subfield $\mathcal{E} = \mathbb{Q}(\beta)$ of $\mathcal{F}$ can be described by a pair (h, g) where g is the minimal polynomial of β and an embedding polynomial $h \in \mathbb{Q}[t]$ with $h(\alpha) = \beta$. In KANT V4 subfields are described in this way and can be computed by a generalized and improved version [Klüners, Pohst (1996)] of Dixon's (1990) method.

Relative Norm Equations Let $\mathbb{Q} \subseteq \mathcal{F} \subseteq \mathcal{E}$ be algebraic number fields and let $M \subseteq \mathcal{E}$ be a free $o_{\mathcal{F}}$ module. For a non–zero $\beta \in o_{\mathcal{F}}$ KANT V4 provides routines for the computation of all non-associate $\gamma \in M$ with $N_{\mathcal{E}/\mathcal{F}}(\gamma) = \beta$. These routines are based on a generalization [Fieker, Jurk, Pohst (1996)] of the algorithm by Fincke and Pohst (1983).

Kummer Extensions Let $\mathcal{F}$ be an algebraic number field containing the n-th roots of unity. We consider an extension $\mathcal{E}/\mathcal{F}$ such that $\mathcal{E} = \mathcal{F}(\sqrt[n]{\mu})$ where $\mu \in o_{\mathcal{F}}$. KANT V4 is able to compute the relative discriminant $\mathfrak{d}_{\mathcal{E}/\mathcal{F}}$ of the extension $\mathcal{E}/\mathcal{F}$ and a small set $\{\xi_1, \ldots, \xi_m\} \subseteq o_{\mathcal{E}}$ such that

$$o_{\mathcal{E}} = \xi_1 o_{\mathcal{F}} + \cdots + \xi_m o_{\mathcal{F}}.$$

Additionally KANT V4 can derive an integral basis of $o_{\mathcal{E}}$ from $\{\xi_1, \ldots, \xi_m\}$ [Daberkow (1995), Daberkow, Pohst (1995), Daberkow, Pohst (1996a)].

Hilbert Class Fields Let $\mathcal{F}$ be an algebraic number field. Making extensive use of computations in Kummer extensions, KANT V4 can compute the Hilbert class field $H(\mathcal{F})$ of $\mathcal{F}$ arithmetically [Daberkow, Pohst (1996b)].

Thue Equations One of the classical objects of number theory is the Diophantine equation of Thue

$$f(X, Y) = a,$$

where $f(X, Y) \in \mathbb{Z}[X, Y]$ is an irreducible form of degree ≥ 3 and a is an integer. In KANT V4 such equations are solved by the methods in Bilu, Hanrot (1995).

4. The shell

A most recent and extremely important part of our software is KASH – the KAnt SHell.

For a proper use of KANT V4, the user needs to have some experience with programming in C and an understanding of the memory management in MAGMA. Because of this disadvantage, we started to build a shell around the C–library KANT V4, which combines the functionality of KANT V4 with a comfortable user interface based on GAP, a software package for group theory [Schönert et al. (1993)].

The interpreter consists of several units, e.g. the KANT V4–package, system–dependent functions, an additional memory manager, an internal function library, etc.

In principle, a simple main–loop is performed:

- READ Reading the command–line from the keyboard, out of a file or from another process (see above: distributed computing).
- EVAL Evaluating the input: the input is tokenized and a multiway tree for evaluation is created. By recursion, the root of the tree is evaluated to a single result (value).
- PRINT Printing the result on the screen, into a file or sending it to another process (see above).

Within the shell, the user can do arithmetical operations with integers, rationals, real and complex numbers (with arbitrary precision), matrices or — after the definition of an order — with algebraic numbers, ideals, etc.. Of course, all results can be assigned to variables for later use.

Furthermore the user can make use of two different kinds of functions, the "internal functions" and the "user functions". The first are built–in functions of the internal function library, i.e. they are written in C, linked to KASH and cannot be changed. With these at hand, most of the algorithms mentioned above can be performed.

In contrast to these, the user can create his own (user) functions: With the PASCAL–like programming language, he can create loops, conditional branches, functions etc. and use all internal and user functions. In this environment, he can even write sophisticated programs. All user functions and programs can be stored as (external) text files which build a user function library (in contrast to the internal function library).

Additionally, KASH posesses an interface to the public domain PVM–software which allows distributed computing (see previous section) and is very easy to handle.

Presently, there are more than 350 internal functions installed, 200 additional predefined user functions and comprehensive references are available. Because KASH grows weekly, updates will be made more often than for the KANT V4–library.

5. Distributed Computing

It is possible, both in KANT V4 and KASH, to benefit from distributed computing. The requirement for this (in addition to KASH) is a network of workstations running PVM3 (at least version 3.3, see [Geist et al. (1994)]). Based on the PVM-protocol we provide a high-level interface for process communication and exchange of KANT V4 data. We support two different modes of communication: one is based on KANT V4, providing C-functions, and the other is based on KASH, consisting of several KASH commands. In the sequel both of these will be described seperately.

Our (virtual) parallel computer is hierarchical, consisting of one master, the KASH session running in the foreground, and an arbitrary number of slaves, running on different machines in the background. If the current task is able to use them, additional slaves can be added at run time. It is possible to give arbitrary time spots for each machine, e.g. allowing the slaves only to run at night or on the weekend, in order to be able to do really large jobs at times which are convenient.

For data security we provide a so called "security system", guaranteeing that all jobs sent will be processed, i.e. if eventually one slave dies (due to network errors, or similar events) it will be restarted (if possible) and all jobs running on that slave will be redistributed among all slaves.

In order to use PVM one has to start it (normally by just typing `pvm`) and to add all hosts one wants to use to the network (e.g. using `add hostname` on the PVM-console, or using a configuration file).

In KANT V4 (that is, using the C-library) and in KASH, distributed computing can be enabled using just one statement, `pvm_set_use_pvm(1);` or in KASH `PvmUse(true);`, provided PVM itself is already started. Afterwards, some packages (e.g. Round-2 or classgroup computation) will use as many slaves as available and convenient.

As an example we discuss the processing of the computation of the maximal order by the Round-2 algorithm. After the factorization of the discriminant, the p-maximal overorders for all primes p whose square divides the discriminant have to be computed. Since those calculations are quite independent, we use different slaves for different primes. The combination of the results is done on the master afterwards. Especially when "large" computations (involving many primes) are carried out, a lot of time can be saved in this way.

As a second example we consider norm equations, which are required for principal ideal tests. Solutions of norm equations are determined as lattice points in suitable ellipsoids. Sometimes it is necessary to enumerate several thousand ellipsoids, especially when there is no solution resp. the ideal is not principal. To save time the ellipsoids can be distributed among all slaves.

Besides these examples, several other algorithms have sub-tasks which can be distributed. When it is possible (i.e. the algorithm permits it and PVM is started and enabled) this will be done automatically without notifying the user.

Note however that it is not advisable to permanently enable distributed computing in this way, since the overhead generated might even increase the computation time for small examples.

In KASH there is another possibility for parallelizing: KASH contains a set of interface functions to PVM, allowing the user to write programs in KASH that utilize PVM to perform certain parts concurrently. The main advantage, in comparison to library calls, are the short developing cycles when writing parallel functions.

6. The database

Accessible from KASH is a SQL–database for number fields [Daberkow, Weber (1996)]. The database is designed to give easy and fast access to several hundreds of thousands of number fields. Currently the following invariants are stored (if known) and can be used as keys in a selection:

a generating polynomial together with its signature
an integral basis, the field discriminant
the unit group and regulator
the class group with structural information
the Galois group

In accordance with PARI we choose a special form of the generating polynomial $f(t) = t^n + a_1 t^{n-1} + \cdots + a_n$ as a unique key for the fields in the database (at least for number fields of low degree). As a generating element we take an algebraic integer ρ subject to the following conditions:

1 $T_2(\rho) = \sum_{j=1}^{n} |\varphi_j(\rho)|^2$ is minimal,

2 the index of the equation order $\mathbb{Z}[\rho]$ in the maximal order is minimal,

3 $a_1 > 0$ is minimal,

4 $|a_i|$ $(2 \leq i \leq n)$ are minimal.

Isomorphy can be tested with KASH. In a first step one can check some invariants and if all tests are successful there is the possibility to choose between several algorithms for proving the isomorphy.

The underlying SQL–database (currently mSQL, Postgres95 interface is under development) is public domain and available for every system supported by KASH.

As an example we will find all totally real cubic fields with discriminant less 10,000 and common inessential discriminant divisors. It is well known that a sufficient and neccessary criterion is that exactly three different prime ideals divide 2, so the following program is straightforward:

```
kash> DbOpen("donald:kantnf");    # open the database
true
kash> query :=                    # we are interested only
> "degree=3 and discriminant<10000 and [number of real zeroes]=3";;
kash>                             # in small totally real cubic fields
kash> DbCountMatchesQueryFLDTable(query);
382
kash> DbQueryFLDTable(query);
true
kash> L:= [];
kash> repeat
>  o := DbNextOrderFromQuery();
>  if o<>false and
>     OrderIndex(o)mod 2 = 0 and  # 2 has to divide the index,
>                                 # this is a fast criterion
>     Length(Factor(2*o))=3 then  # exactly 3 different primes
>     Add(L, o);
>  fi;
>  until o=false;
kash> Length(L);                  # we found 14 fields
14
```

7. Examples

In the sequel we give some examples of KASH.

7.1. Computing the maximal order, unit group and class group

We start with the equation order $\mathbb{Z}[\rho]$ for $\rho^4 - 117 = 0$ and compute a set of fundamental units.

First we create the order $\mathbb{Z}[\sqrt[4]{117}]$ of degree 4 over $\mathbb{Z}$.

```
kash> o := Order(Z,4,117);
Generating polynomial:  x^4 - 117
```

We compute the fundamental units in the equation order. Setting $\rho = \sqrt[4]{117}$ the first fundamental unit is $649 - 60\rho^2$.

```
kash> OrderUnitsFund(o);
[ [649, 0, -60, 0], [26618086, -8093388, 2460843, -748234] ]
```

To calculate the index of the unit group of the equation order in the unit group of the corresponding maximal order, we proceed as follows.

```
kash> O := OrderMaximal(o);
   F[1]
    |
   F[2]
  /
 /
Q
F  [ 1]     Given by transformation matrix
F  [ 2]     x^4 - 117
Generating polynomial:  x^4 - 117
Discriminant: -316368
```

A transformation matrix from a basis of `o` to a basis of `O` is stored but not printed (it can be obtained with the command `OrderTransformationMatrix`).

```
kash> OrderUnitsFund(O);
[ [2, 0, -1, 0], [1, -1, 1, 0] ]
```

The units are representated in the basis of the maximal order. After determining a set of fundamental units, we get the index as the quotient of the two regulators.

```
kash> OrderReg(o)/OrderReg(O);
36
```

Finally, we calculate the class group structure:

```
kash> OrderClassGroup(O);
[ 4, [2, 2] ]
```

This means that the class group is of order 4 and is isomorphic to $C_2 \times C_2$.

7.2. Computing subfields

The following example demonstrates the computation of subfields: We start by creating the equation order $\mathbb{Z}[\rho]$ for $\rho^6 + 108 = 0$.

```
kash> o:=Order(Z,6,-108);
Generating polynomial:  x^6 + 108
```

The computation of proper subfields of the quotient field $\mathbb{Q}(\rho)$ of `o` yields the following list of equation orders.

```
kash> L:=OrderSubfield(o);
[ Generating polynomial:  x^3 - 108
    , Generating polynomial:  x^3 - 108
    , Generating polynomial:  x^3 - 108
    , Generating polynomial:  x^2 + 108
     ]
```

There are 3 subfields of degree 3 which are isomorphic but not identical and one subfield of degree 2. Let ρ_1, ρ_2, ρ_3 denote the roots of $x^3 - 108$. L[i] denotes the i-th equation order in L.

```
kash> r1:=Elt(L[1],[0,1,0]);
[0, 1, 0]
kash> r2:=Elt(L[2],[0,1,0]);;
kash> r3:=Elt(L[3],[0,1,0]);;
```

The elements look identical, but they are indeed different which is detected upon moving them into the order o.

```
kash> EltMove(r1,o)          # This produces the element
                             # (6*rho^2-rho^5)/12
[0, 0, 6, 0, 0, -1] / 12
kash> EltMove(r2,o);         # This produces the element -rho^2
[0, 0, -1, 0, 0, 0]
kash> EltMove(r3,o);         # This produces the element
                             # (6*rho^2+rho^5)/12
[0, 0, 6, 0, 0, 1] / 12
```

Any element of L[i] can be lifted in an analogous way.

7.3. Solution of Thue–equations

Given an irreducible form $f \in \mathbb{Z}[X,Y]$ of degree ≥ 3 and an integer a, we compute all $(x,y) \in \mathbb{Z}^2$ subject to $f(x,y) = a$.

Let $f(X,Y) := X^3 + X^2Y - 6XY^2 + 2Y^3$ and solve $f(x,y) = 2$. The corresponding number field $\mathcal{F}$ is created by a root of the irreducible polynomial $f(X,1) \in \mathbb{Z}[X]$.

```
kash> t := Thue([1,1,-6,2]); # [1,1,-6,2] are the coefficients of f.
X^3 + X^2 Y - 6 X Y^2 + 2 Y^3
```

```
kash> Solve(t,2);            # Compute a list of all solutions [x,y].
[ [ -724, -411 ], [ -4, -11 ], [ -3, 1 ], [ -1, -1 ],
  [ 0, 1 ], [ 2, 1 ] ]
```

Additionally, we can solve Thue–equations up to sign on the right hand side, for example $X^7 + X^6Y - 6X^5Y^2 - 5X^4Y^3 + 8X^3Y^4 + 5X^2Y^5 - 2XY^6 - Y^7 = \pm 1$.

```
kash> t := Thue([1,1,-6,-5,8,5,-2,-1]);
X^7 + X^6 Y - 6 X^5 Y^2 - 5 X^4 Y^3 + 8 X^3 Y^4 + 5 X^2 Y^5\
 - 2 X Y^6 - Y^7
kash> Solve(t,1,"abs");      # Compute a list of all solutions [x,y].
[ [ -2, -1 ], [ -1, -1 ], [ -1, 0 ], [ -1, 1 ], [ 0, -1 ],
  [ 0, 1 ], [ 1, -1 ], [ 1, 0 ], [ 1, 1 ], [ 2, 1 ] ]
```

7.4. Solution of norm equations

We consider the relative extension $\mathcal{E}/\mathcal{F}$ for

$$\mathcal{F} = \mathbb{Q}(\alpha),\ \alpha^2 - 2 = 0, \quad \text{and} \quad \mathcal{E} = \mathcal{F}(\beta),\ \beta^2 + 1 = 0.$$

For all $\theta = i\alpha + j$ $(-4 \leq i, j \leq 4)$ we want to know if there exists $\mu \in o_{\mathcal{E}}$ with $\mathrm{N}_{\mathcal{E}/\mathcal{F}}(\mu)/\theta \in \mathrm{TU}_{\mathcal{F}}$ the group of torsion units.

```
kash> F := Order(Z, 2, 2);;
kash> E := Order(F, 2, -1);;
kash> l1:=[];                # Create an empty list.
kash> zero := Elt(F, 0);;    # Create the zero element in F
kash> for i in [0..4] do
>       for j in [-4..4] do
>         z := Elt(F, [i,j]);
>         if z<>zero then
>           Add(l1, [z, OrderNormEquation(E, z)]);
>         fi;
>       od;
>     od;
```

The output consists of a list containing pairs $[\theta, \mu]$ if there is a solution μ and $[\theta,$`false`$]$ otherwise (optional).

```
[ [ [0, -4], false ], [ [0, -3], false ],
  [ [0, -2], false ], [ [0, -1], false ],
  [ [0, 1], false ], [ [0, 2], false ], [ [0, 3], false ],
...
  [ [1, -1], false ], [ 1, [ [[0, -1], [0, 1]] / 2 ] ],
...
  [ [4, 3], false ], [ [4, 4], false ] ]
```

It took 17 seconds to solve these 44 norm equations.

7.5. Computation of Hilbert class fields

The following is an example for the computation of the Hilbert class field for $\mathcal{F} := \mathbb{Q}(\rho)$ where $\rho^3 + \rho^2 - 42\rho - 107 = 0$.

We start by reading the equation order of $\mathcal{F}$.

```
kash> f:=Poly(Zx,[1,1,-42,-107]);;  # f(x)=x^3+x^2-42x-107
kash> F:=Order(f);;                 # Create the equation order
                                    # of the polynomial f
kash> F:=OrderMaximal(F);;
kash> OrderUnitsFund(F);;
kash> OrderClassGroup(F,"euler");;  # option "euler" is necessary
                                    # for the function
                                    # OrderHilbertClassField
kash> F;
Generating polynomial:  x^3 + x^2 - 42*x - 107
Discriminant: 70313
Regulator: 21.20506
Units:
[3, 1, 0]       [9, 12, 2]
class number 2
class group structure C2
cyclic factors of the class group:
<5, [3, 0, 1]>
```

The discriminant is always the discriminant of the order. The cyclic factors are given in a 2 element normal representation.

We apply the user function `OrderHilbertClassField` to it.

```
kash> Y := OrderHilbertClassField (F);
Starting Class Field Computation
   Degree      : 3
   Signature   : [ 3, 0 ]
   Class Group : [ 2, [ 2 ]]
=====================================
Checking cyclic group C2
----------------------------------------------------
----------------------------------------------------
Computing Classfield for cyclic subgroup C2
```

We obtain the following 4 elements $\alpha_1, \dots, \alpha_4$, a power product of which yields a generating element.

```
List of Generators :
[ [1299, 255, -62], -1, [3, 1, 0], [9, 12, 2] ]
```

We compute a generating element $\mu = \alpha_1^{e_1} \cdot \ldots \cdot \alpha_4^{e_4}$ for $(e_1, \dots, e_4) \in (\mathbb{Z}/2\mathbb{Z})^4$. Only unramified extensions $\mathcal{F}(\sqrt[2]{\mu})$ of $\mathcal{F}$ are processed further.

```
Exponent Vector [ 1, 1, 0, 1 ] -->[79, 5, -2]
```
That means $\mu = 79 + 5\rho - 2\rho^2$.

Since we obtain just one unramified extension, it has to be the class field. There is also a built in checking routine, however.

8. Availability

KASH is freely available via `ftp.math.tu-berlin.de` at `pub/algebra/Kant/Kash`. It has been ported to the following architectures:

HP 7000 (HP–UX 9.01),
IBM RS 6000 (AIX 3.2.5),
Intel 486 (Linux Kernel 1.3.0),
Intel 486 (MS DOS 5.0),
Silicon Graphics (IRIX 5.3),
Sun SPARC (SunOS 4.1.3),
Sun SPARC (SunOS 5.4).

References

Bilu, Y.; Hanrot, G.: *Solving Thue Equations of High Degree*; Mathematica Gottingensis, Schriftenreihe des Sonderforschungbereichs Geometrie und Analysis; Heft 11; Göttingen; 1995

Cannon, J.: *Magma*; this volume

Cohen, H.: *A Course in Computational Algebraic Number Theory*; Springer-Verlag; Berlin - Heidelberg - New York; 1993

Daberkow, M.: *Über die Berechnung der ganzen Elemente in Radikalerweiterungen algebraischer Zahlkörper*; Thesis; Berlin; 1995

Daberkow, M.: *Computing with Subfields*; submitted to J. Symb. Comput.

Daberkow, M.; Pohst, M.: *Computations with relative extensions of number fields with an application to the construction of Hilbert class fields*; ISSAC'95 Proc.

Daberkow, M.; Pohst, M.: *On Integral Bases in Relative Quadratic Extensions*; to appear in Math. Comput., 1996a

Daberkow, M.; Pohst, M.: *On the Computation of Hilbert Class Fields*; submitted for publication, 1996b

Daberkow, M.; Weber, A.: *A Database for Number Fields*; submitted to DISCO'96

Dixon, J.: *Computing Subfields in Algebraic Number Fields*; J. Aust. Math. Soc., Ser. A 49; 1990; 434–448

Eichenlaub, Y.; Olivier, M.: *Computation of Galois groups for polynomials with degree up to eleven*; to appear

Fieker, C.; Jurk, A.; Pohst, M.: *On Solving Relative Norm Equations in Algebraic Number Fields*; to appear in Math. Comput.

Fincke, U.; Pohst, M.: *A Procedure for Determining Algebraic Integers of Given Norm*; Proceedings EUROCAL 83; Springer LNCS Series No. 162; Springer-Verlag; Berlin - Heidelberg - New York; 1983

Geist, A. et al.: *PVM 3 user's guide and reference manual*; Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831; 1994

Klüners, J.; Pohst, M.: *On Computing Subfields*; submitted to J. Symb. Comput.

Pohst, M.: *Computational Algebraic Number Theory*; Birkhäuser; Basel; 1993

Pohst, M.; Zassenhaus, H.: *Algorithmic Algebraic Number Theory*; Cambridge University Press; Cambridge; 1989

Schönert, M. et al.: *GAP – Groups, Algorithms and Programming, version 3 release 3*; RWTH Aachen, Lehrstuhl D für Mathematik; Templergraben 64, 52062 Aachen, F.R.G.; 1993