FIVE-DIMENSIONAL K-OPTIMAL LATTICE RULES

J. N. LYNESS AND TOR SØREVIK

ABSTRACT. A major search program is described that has been used to determine a set of five-dimensional K-optimal lattice rules of enhanced trigonometric degrees up to 12. The program involved a distributed search, in which approximately 190 CPU-years were shared between more than 1,400 computers in many parts of the world.

Subject Classification: Primary: 41A55, 41A63, 42A10; Secondary: 65D32

Keywords: Multidimensional cubature, optimal lattice rules, K-optimal rules, and optimal trigonometric rules.

1. INTRODUCTION

1.1. **Historical Background.** A lattice rule $Q(\Lambda)$, is a uniform weight numerical integration rule, for the evaluation of integrals over an s-dimensional unit cube, which takes as its abscissas those points of an s-dimensional *integration* lattice Λ that lie within the unit cube (see (2) below).

Associated with each lattice rule is its *abscissa count*, denoted by $N(\Lambda)$; this coincides with the inverse of the order (or density) of the lattice Λ.

Lattice rules are particularly effective when the integrand function is periodic with respect to the unit cube and has a high degree of continuity. Consequently, since their inception, interest has focused on their trigonometric degree.

An integration rule is said to be of *enhanced* trigonometric degree δ when it integrates all trigonometric polynomials of degree $\delta - 1$ exactly.

Let $\mathcal{L}$ stand for a subset of s-dimensional lattice rules. An $\mathcal{L}$*-optimal* rule of enhanced degree δ is a rule belonging to this subset (population) of this enhanced degree having minimal abscissa count.

Noskov and his collaborators [Nos91] were the first to publish a set of s-dimensional optimal lattice rules of enhanced degree 4 for all s. (See also [Mys88].) At the same time, they published some higher-degree sequences of rules. However, they did not exploit the properties of the dual lattice described later in this paper, and they confined their treatment to rank 1 rules. These rules are far from optimal. Later work ([LySø91], [LySø92], [LySø93],[CoLy01],[LySø04] and [CoGo03]) has exploited the elegant theory involving the dual lattice.

Nearly all results about moderate-dimensional rules of moderate degree appear to have been obtained by means of numerical searches. For larger s and δ, the search

The first author's work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-Eng-38.

populations become outrageously large. Usually the search has to be confined to smaller populations; and the resulting rules, occasionally the most economic then available, cannot be expected to be optimal.

In spite of this situation, significant intellectual and computational effort has been devoted to finding optimal lattice rules. The present paper describes another contribution to that effort. Here we follow a path first described by Cools and Lyness [CoLy01], where the search is confined to a well-defined subset of all lattice rules, namely, $\mathcal{K}(s,\delta)$, defined in Section 2. There we recall their definition of the set $\mathcal{K}(s,\delta)$ and describe the basic theory, which, in a five-dimensional context, involves *quintets* of *facet-pairs*.

In Section 3 we describe salient features of our search algorithm. In Section 3.1, we describe how to exploit some of the symmetry properties of quintets to reduce significantly the redundancy in the search population. In Section 3.2 we describe how to apply previously available bounds on N and δ to limit the search space.

In Section 4 we briefly describe how to parallelize the code and run it as a distributed search on Internet-connected computer systems. In section 5 we describe the behavior of the program.

In Section 6 we present the results of the calculation. These comprise nine new five-dimensional K-optimal lattice rules of enhanced degree between 5 and 12. For these values of δ, these five-dimensional rules have a lower abscissa count than any other published rule known to the authors.

1.2. **Background on Integration Lattices and Their Trigonometric Degree.** An s-dimensional lattice Λ may be defined in terms of a set of s linearly independent s-dimensional points, termed generators. Any linear, integer combination of these generators is a lattice point. Any $s \times s$ matrix A, whose s rows coincide in some order with any set of generators, is called a generator matrix of the lattice. This lattice may be defined by

$$\mathbf{x} \in \Lambda \Leftrightarrow \mathbf{x}^T = \lambda^T A; \quad \text{where } \lambda \in \Lambda_0. \tag{1}$$

Here, Λ_0 is the unit lattice, that is the lattice consisting of all integer points. An *integration lattice* is one that contains the unit lattice. Thus it includes all 2^s vertices of the integration region $[0,1)^s$. Only integration lattices may be used to construct lattice rules.

$$If = \int_{[0,1)^s} f(\mathbf{x})d\mathbf{x}; \quad Q(\Lambda)f = \frac{1}{N(\Lambda)} \sum_{\mathbf{x}\in\Lambda\cap[0,1)^s} f(\mathbf{x}). \tag{2}$$

The matrix $B = (A^{-1})^T$ is a generator matrix for another lattice, which is called the dual lattice of Λ, and is denoted by $\Lambda^\perp$. When A defines an integration lattice, the elements of A are rational numbers, and B is an integer matrix (one whose elements are integers)[Lyn89]. The dual lattice plays a major role in the theory of lattice rules. For example, the discretization error Ef has the expansion

$$Ef = If - Q(\Lambda)f = \sum_{\substack{\mathbf{h} \in \Lambda^\perp \\ \mathbf{h} \neq 0}} \hat{f}(\mathbf{h}), \tag{3}$$

where $\hat{f}(\mathbf{h})$ is a Fourier coefficient of the integrand function $f(\mathbf{x})$.

An integration rule is said to be of trigonometric degree d when it integrates all trigonometric polynomials of degree d exactly [CoSl96]. Its *enhanced* degree is defined as $\delta = d + 1$. It follows from (3) that the *strict* enhanced degree of $Q(\Lambda)$ is

$$\delta(\Lambda) = \min_{\substack{\mathbf{h} \in \Lambda^{\perp} \\ \mathbf{h} \neq 0}} \| \mathbf{h} \|_1 . \tag{4}$$

It is also well known (see, e.g., [Lyn89]) that the number of abscissas required by this lattice rule is

$$N(\Lambda) = | \det(A) |^{-1} = | \det(B) | . \tag{5}$$

Every nonsingular integer matrix B defines a dual lattice, and every integer lattice has at least one nonsingular generator matrix B. Consequently both N and δ above may be considered to be uniquely defined functions of B. In cases where no confusion is likely to arise, we condense the notation to reflect this. For example, by $\delta(B)$ we mean the value given by (4) where $\Lambda^{\perp}$ is the lattice generated by B. Finding an optimal lattice rule of enhanced degree δ involves finding a solution to the minimization problem

$$N_{opt}(s, \delta) = \min_{\forall B;\ \delta(B) = \delta} | \det(B) | . \tag{6}$$

2. The set $\mathcal{K}(s, \delta)$ and the basic structure of the search

We define $\bar{\Omega}(s, \delta)$, a set of points all of which lie on the boundary of an s-dimensional octahedron, as follows:

$$\mathbf{h} \in \bar{\Omega}(s, \delta);\ ||\mathbf{h}||_1 = \sum_{i=1}^{s} |h_i| = \delta \textit{ and } \mathbf{h} \in \Lambda_0. \tag{7}$$

In view of (4), any lattice of enhanced degree δ may have only one point, namely the origin, inside this octahedron. Intuititively one might expect an optimal lattice to have a preponderance of points near or on the boundary of this octahedron. This configuration is discussed in some detail in [CoLy01]. Every member of the set $\mathcal{K}(s, \delta)$ defined below has one or more generator matrices comprising s generators on $\bar{\Omega}(s, \delta)$.

The three-dimensional octahedron has eight plane faces, or four pairs of opposite faces. More generally, the set $\bar{\Omega}(s, \delta)$ is composed of 2^s distinct $(s-1)$-dimensional facets or 2^{s-1} distinct facet-pairs. Each facet contains $\begin{pmatrix} s + \delta - 1 \\ s - 1 \end{pmatrix}$ lattice points. There is minor overlap between neighbouring facets.

Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_s)$, where each σ_i is either $+1$ or -1. These 2^s distinct values of σ can be used to specify one of the 2^s facets as follows.

Definition 2.1. *([CoLy01]) Let δ be a positive integer. The* facet-pair $F(s, \delta, \sigma)$ *comprises two facets of $\bar{\Omega}(s, \delta)$. One contains all* $\mathbf{h}$ *satisfying*

$$\mathbf{h} \in \bar{\Omega}(s, \delta) \tag{8}$$

and

$$h_i = \sigma_i |h_i|;\ \forall i = 1, 2, \ldots, s. \tag{9}$$

The other is the opposite facet for which

$$h_i = -\sigma_i |h_i|;\ \forall i = 1, 2, \ldots, s. \tag{10}$$

In the five-dimensional case, there are precisely 16 distinct facet-pairs. These are listed and assigned a serial number in Table 1.

TABLE 1. **Facet-pairs:** The sign pattern of the coordinates of points on each of the 16 five-dimensional facet-pairs.

F_0	$F(s,\delta,+,+,+,+,+)$
F_1	$F(s,\delta,-,+,+,+,+)$
F_2	$F(s,\delta,+,-,+,+,+)$
F_3	$F(s,\delta,-,-,+,+,+)$
F_4	$F(s,\delta,+,+,-,+,+)$
F_5	$F(s,\delta,-,+,-,+,+)$
F_6	$F(s,\delta,+,-,-,+,+)$
F_7	$F(s,\delta,-,-,-,+,+)$
F_8	$F(s,\delta,+,+,+,-,+)$
F_9	$F(s,\delta,-,+,+,-,+)$
F_{10}	$F(s,\delta,+,-,+,-,+)$
F_{11}	$F(s,\delta,-,-,+,-,+)$
F_{12}	$F(s,\delta,+,+,-,-,+)$
F_{13}	$F(s,\delta,-,+,-,-,+)$
F_{14}	$F(s,\delta,+,-,-,-,+)$
F_{15}	$F(s,\delta,-,-,-,-,+)$

We are now in a position to define the set $\mathcal{K}(s,\delta)$ as the set of all lattices generated by five points each of which is a member of a *distinct* facet-pair. Since any lattice in $\mathcal{K}(s,\delta)$ contains points on $\bar{\Omega}(s,\delta)$, its enhanced degree cannot exceed δ and is generally less than δ. A K-optimal lattice may now be defined as follows.

Definition 2.2. *A lattice in $\mathcal{K}(s,\delta)$ with enhanced degree equal δ having the minimum abscissa count is said to be a K-optimal lattice rule.*

There are many ways of choosing the five distinct facet-pairs. A *quintet*, defined below, defines a specified selection of facet-pairs.

Definition 2.3. *A* quintet*, denoted by $q(s=5,\ \delta;\ N_1,\ N_2,\ N_3,\ N_4,\ N_5)$, where $0 \le N_1 < N_2 < N_3 < N_4 < N_5 \le 15$, comprises the set of the five distinct facet-pairs F_{N_1},F_{N_2},F_{N_3},F_{N_4} and F_{N_5}.*

In the sequel, we suppress the arguments s,δ where no confusion is likely to arise, and we use the abbreviation $\mathbf{N} = (N_1, N_2, N_3, N_4, N_5)$.

A lattice is said to *belong* to a quintet q if it can be generated by five points, each of which belongs to a different facet-pair of that quintet. We denote by $\mathcal{Q}(s,\delta;\mathbf{N})$, the set of lattices belonging to $q(s,\delta;\mathbf{N})$.

Every lattice in $\mathcal{K}(s,\delta)$ must belong to at least one quintet. The same lattice may, however, belong to many different quintets. There are 4,368 quintets. Eleven of these are listed in the central column of Table 2.

Naturally the union of all $\mathcal{Q}(s,\delta,\mathbf{N})$ coincides with $\mathcal{K}(s,\delta)$.

3. Restricting the search space

The scope of the search can be reduced in two different ways. First, symmetry properties can be used to reduce the total number of *quintets* it is necessary to search. Second our search can be organized as a branch-and-bound routine, using the constraint, $\delta(B) = \delta$ to omit subspaces where $\delta(B) < \delta$. For the remaining lattices we describe efficient computation of $N(B)$ and how to take advantage of upper and lower bounds on $N(B)$ to reduce the number of times $\delta(B)$ has to be computed.

3.1. **Equivalence Classes of quintets.** Let $\mathcal{G}$ stand for the group of rotations and reflections of the axes that takes the five-dimensional unit hypercube into itself. Let G be a standard 5×5 integer matrix representing one of the group operations in $\mathcal{G}$.

The set of lattices obtained from a specified lattice, Λ, by rotation and reflection using individual members of the group, $\mathcal{G}$ are said to be *symmetric copies* of each other, and collectively they constitute an *equivalence class*. Since $\mathcal{G}$ is a group of order 3840, there may be up to 1920 members in a particular equivalence class.

Generator matrices B and $\tilde{B}$ of the lattices $\Lambda(B)$ and $\Lambda(\tilde{B})$, in the same equivalence class are related by the equation

$$\tilde{B} = UBG, \tag{11}$$

where U is a unimodular matrix. As always, premultiplication of B by U changes only the generator matrix, not the lattice.

When B and $\tilde{B}$ are in the same equivalence class, it follows that

- $N = \mid \det(B) \mid = \mid \det(\tilde{B}) \mid$, and
- $\delta(B) = \delta(\tilde{B})$.

Moreover,

- if $B \in \mathcal{K}(s, \delta)$, then $\tilde{B} \in \mathcal{K}(s, \delta)$.

In view of these properties the following lemma enables us to greatly reduce the scope of the search.

Lemma 3.1. *If a lattice $\Lambda \in \mathcal{Q} = \mathcal{Q}(s, \delta, (N_{i_1}, N_{i_2}, N_{i_3}, N_{i_4}, N_{i_5}))$ has a symmetric copy $\tilde{\Lambda} \in \tilde{\mathcal{Q}} = \mathcal{Q}(s, \delta, (N_{j_1}, N_{j_2}, N_{j_3}, N_{j_4}, N_{j_5}))$, then every lattice in $\mathcal{Q}$ has a symmetric copy in $\tilde{\mathcal{Q}}$.*

Proof. Let B and $\tilde{B}$ be generator matrices for Λ and $\tilde{\Lambda}$, constructed by points on the five facets $N_{i_1}, N_{i_2}, N_{i_3}, N_{i_4}, N_{i_5}$ and $N_{j_1}, N_{j_2}, N_{j_3}, N_{j_4}, N_{j_5}$, respectively. Then if Λ and $\tilde{\Lambda}$ belong to the same equivalence class, there exists a $G \in \mathcal{G}$ such that $\tilde{B} = UBG$. Since by definition every lattice in $\mathcal{Q}(s, \delta, q(N_{i_1}, N_{i_2}, N_{i_3}, N_{i_4}, N_{i_5}))$ has a generator matrix constructed by points from the specified quintet, applying the same group operations to any of these generator matrices will produce a generator matrix for a symmetric copy in the quintet $q(N_{j_1}, N_{j_2}, N_{j_3}, N_{j_4}, N_{j_5})$. ∎

The vital consequence of this lemma is that the quintets can be assembled into equivalence classes and that it is sufficient to examine the lattices of only one quintet in each equivalence class in the search for N_{opt} of $\mathcal{K}(s, \delta)$.

Theorem 3.1. *For $s = 5$ there are exactly 11 distinct equivalence classes of quintets.*

TABLE 2. **Leading quintets:** The 11 distinct equivalence classes of quintets are identified by their *leading quintet*, given in the second column. In the final column is the number of distinct quintets in that equivalence class. The facet-pairs are defined in Definition 2.1 and numbered in Table 1.

q_0:	$q(0,1,2,3,4)$	480
q_1:	$q(0,1,2,4,7)$	160
q_2:	$q(0,1,2,4,8)$	80
q_3:	$q(0,1,2,4,9)$	960
q_4:	$q(0,1,2,5,6)$	480
q_5:	$q(0,1,2,5,10)$	192
q_6:	$q(0,1,2,5,11)$	960
q_7:	$q(0,1,2,7,11)$	480
q_8:	$q(0,1,6,7,10)$	240
q_9:	$q(0,1,6,10,12)$	320
q_{10}:	$q(0,3,5,9,14)$	16

One quintet of each equivalence class is listed in Table 2. These are the *leading quintets* in a natural lexicographical ordering. For these we introduce the shorthand notation $q_i;\ i = 0, \cdots, 10$.

Proof. The theorem can be established in a constructive way simply by carrying out a sequence of calculations using the following straightforward algorithm. We set up a list of the 4,368 *quintets* and apply in turn all 1,920 symmetry transformations to the first member of the list, removing from the list all quintets constructed this way. We take next the first untreated quintet and repeat the process; we continue in this way until no more untreated quintets remain on the list. The resulting list contains only the 11 quintets listed in Table 2. We have programmed this tedious, trivial, and time-consuming computation; the result is reported in the theorem and in Table 2. □

The corresponding result for quartets is 4, given in [CoLy01] and for sextets is 131. These, together with the theorem above, were established using a more sophisticated and efficient program constructed by Dr Cools in 1998.

Restricting our search to only one *quintet* of each equivalence class reduces the number of quintets treated from the original 4,368 to 11 and may reduce the overall cost by a factor of roughly 400.

While running the large search program (described later), we found that for $\delta \geq 8$ it became necessary to restrict drastically the search space. We did this in a somewhat arbitrary way. We reduced the search population from the eleven quintets (listed in Table 2) to a subset of one of those quintets.

The symbol q_3^+ denotes the subset of $q_3 = q(5,\delta;0,1,2,4,9)$ that includes all of F_1, F_2, F_4 and F_9, but only those points of F_0 that satisfy $0 \leq x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq \delta$.

Definition 3.1. *A lattice in q_3^+ with enhanced degree equal δ having the minimum abscissa count, is said to be a K^+-optimal lattice rule.*

3.2. **The Search Algorithm.** Our search space is over all matrices B, representing lattices belonging to all the eleven leading quintets. We organize our search space in a tree structure. At level 1 we have the different points on facet N_1. At level 2 are their children, all combinations of points on facet N_1 and N_2; subsequent levels are organized in this same way. Each leaf at level 5 then has 5 points taken from the 5 prescribed facets that constitute a candidate lattice, B. A subtree, rooted at level k, contains all generator matrices with a fixed combination of the first k rows.

Since they belong to $K(s;\delta)$, all our candidate lattices have $\delta(B) \leq \delta$. But we are interested only in those having $\delta(B) = \delta$. Thus, when treating a particular node, if we can show that all B in the subtree rooted in this node have $\delta(B) < \delta$, the entire subtree can be discarded. It follows from (4) that $\delta(B) \leq \delta(\mathbf{h}) = ||\mathbf{h}||_1$ for all $\mathbf{h} \in \Lambda$, other than the origin. In practice, at a level k node we evaluate $\delta(\mathbf{h}) = ||\mathbf{h}||_1$ for each of $\mathbf{h} = \mathbf{b}_i \pm \mathbf{b}_j$; $1 \leq i < j \leq k$, discarding all matrices B in the subtree rooted at this node if any $\delta(\mathbf{h}) < \delta$ is encountered.

This cut-off strategy enables us to avoid at a relatively low cost, the task of examining the majority of the lattices. The time-consuming part of the program is concerned with the large number of remaining lattices.

As the computation proceeds, we establish and maintain a list of *best-so-far* lattices. The list contains all lattices encountered having abscissa count N_{bsf}, the minimum N of all lattices examined so far.

For a lattice to enter the *best-so-far* list, the two criteria $N(B) \leq N_{bsf}$ and $\delta(B) = \delta$ need to be satisfied. If B fails one of these criteria, the quantity needed for the second criterion, need not to be computed. Computing $\delta(B)$ is much more expensive than computing $N(B)$. We first calculate $N(B)$, and only if this is acceptable need we compute $\delta(B)$.

There exist also lower bounds on $N(B)$ for lattices of given dimension s and enhanced degree δ. Thus, if we find $N(B) < N_{low}(s,\delta)$, we can conclude that $\delta(B) < \delta$ and can omit the computation of $\delta(B)$.

A simple lower bound is based on the well-known classical Minkowski [Min11] result for all lattices, which in this context is

$$N > N_{lat}(s,\delta) = \delta^s/s! \tag{12}$$

In a later version of this program we used the Möller-Mysovskikh bound [Möl79] and [Mys88]. This is based on the underlying set of moment equations and is valid for all rules of enhanced trigonometric degree δ.

$$N \geq N_{ME}(5,\delta) = \begin{cases} \delta(\delta^4 + 20\delta^2 + 24)/120; & \delta \textit{ even} \\ \delta(\delta^4 + 30\delta^2 + 89)/120; & \delta \textit{ odd} \end{cases} \tag{13}$$

The set of B matrices on leaves descending from the same parent have the same first 4 rows of their B matrix. This can be exploited as follows. At level 4 we compute the five 4×4 minors (m_j; $j = 1, \cdots, 5$), and at level 5 we evaluate $N(B)$ using a cofactor expansion

$$N(B) = |\sum_{j=1}^{5} (-1)^j b_{5,j} m_j|. \tag{14}$$

In our context, this classical method turns out to be more efficient than using, many times over, a currently standard algorithm for computing a stand-alone determinant.

4. A distributed search

We have parallelized our search in a conventional server-client style. We have one server that sets up tasks and distributes them to the clients, keeping track of what has already been completed and, of course, retaining the *best-so-far* list.

A single *task* involves processing one subtree rooted at level 2; the input for the client is just the two first rows of a generator matrix B, the sign pattern to be used in constructing the next three rows of B, and the current value of N_{bsf}. The client is to examine each choice of the last three rows, following a set program based on the algorithm outline in the previous section. When this has been done, the client either reports a single possibly optimal lattice (one for which $N \leq N_{bsf}$), or reports that it has found none.

The choice of using subtrees rooted at level 2 as our task unit is a pragmatic one. We have found this results in a good balance between having enough tasks to distribute, making the algorithm scalable, and having tasks large enough for the computation time to dominate the communication time. In particular, more communication could potentially create a bottleneck at the server, which is involved in all the communication that take place.

Avoiding a bottleneck at the server is crucial for the scaling of our algorithm; naturally, we must avoid a situation in which clients are kept waiting in line for the server and are unable to carry out useful computation. For this reason we have minimised the amount of communication per task. Information on quintets, facet-pairs, and node description are coded as indices. The clients have lookup tables corresponding to Table 1 and Table 2 from which they construct the necessary information on quintets and facet-pairs. In order to find the two upper rows in the B matrix for given indices, another list of a similar nature is used. These lists are not communicated. They reside with the client. The short vector of indices that is communicated is used to locate more extensive data items by the client.

This coding practice of communicating by transmitting only a short list of integers to the client minimizes communication at the cost of minor extra work by the clients. The content of the information transmitted is, however, unaltered. At one point we have admittedly sacrificed information to keep the communication slim. When a client finds several potential candidates for our *best-so-far* list, only one is reported back to the server. This approach reduces not only the communication but also the subsequent work in storing and maintaining the list at the server.

The time taken to carry out a single task varies significantly. In an extreme case where the task can be discarded already at level 2 because $\delta(\mathbf{h}) < \delta$ for $\mathbf{h} = \mathbf{b}_1 - \mathbf{b}_2$ the entire task is completed in milliseconds. Generally, the computation may take several seconds for $\delta = 5$ up to several hours for larger δ.

Since so little data is communicated between the server and the client, in particular for larger δ, this is a good example of a coarse-grained parallelism, well suited for distributed computing on a low-speed, high-latency network.

Hence we organized our program as a distributed search, and in July 1999 a general invitation was issued to all people with Internet access to take part. Participation was effected by visiting the Web page, `http://www.ii.uib.no/grisk` and downloading the client program. After being unpacked and started, the client contacts the server and receives a task description. After completing the computation of the task, the client reports the result to the server and is assigned a new

TABLE 3. Summary of the number of tasks computed and the overall CPU-time used. The three central columns are discussed in the text. *This is the number of tasks computed when the program was terminated. The full search over $\mathcal{K}(s, 12, q_3^+)$ contains 85,540 tasks.

	Search over $\mathcal{K}(s,\delta)$				
δ	$T(\delta)$	$\nu_{bsf}(\delta)$	$\nu(\delta)$	$\mu(\delta)$	CPU-time
5	15876	4643	1414	1	6 days
6	44100	16966	960	1	179 days
7	108900	6558	1885	1	5.5 years
8	245025	31798	2400	2	67 years
	Search over $\mathcal{K}(s,\delta,q_3^+)$				
	$T(\delta,q_3^+)$	$\nu_{bsf}(\delta,q_3^+)$	$\nu(\delta,q_3^+)$	$\mu(\delta,q_3^+)$	CPU-time
7	4290	160	117	1	-
8	8910	756	304	2	47 days
9	16445	15	11	1	261 days
10	30030	127	83	1	4.8 years
11	50505	10	10	1	36 years
12	63101*	207	120	1	76 years

task. A detailed description on the technical details of the implementation is given in [SøMy01].

In August 2003, after more that four years, the server was taken out of service, and the program was terminated. It had consumed in total the equivalent of approximately 190 CPU-years spread across more than 1,400 computers.

The quoted CPU-time is simply the sum of individual CPU-times as reported by different clients using individual measuring programs. This crude measure is included here only to give an idea of the order of magnitude of the overall computation. It is not a suitable number to relate to system efficiency.

We relied on the cooperation of many clients who voluntarily contributed time and cycles. We appreciate their time, and we thank them. This situation did have an effect on the programming. As mentioned above, we were restricted to minimal memory. And the program was run at lowest priority. But a major consideration was that the program should need no attention once it was started. Naturally, in running any program, one notices places where minor improvements are clearly in order. (In our case, one was the choice of N_{low} mentioned above. Another involved an extension of the *best-so-far* list discussed in the next section.) Improvements of this type were not initiated.

5. The performance of the search

As δ increases, the computational complexity increases sharply. With our present code it took roughly a year to complete the search over $K(5;\delta)$ with $\delta = 8$, even with hundreds of workstations at our disposal. To obtain any results for higher values of δ, we were forced to use a much smaller search population, namely, $K(5;\delta;q_3^+)$. (See Definition 3.1).

In Table 3 we provide some statistics about the search.

$T(\delta)$ is the total number of *tasks*, and $\nu_{bsf}(\delta)$ is the number of items on the *best-so-far* list when the calculation for that value of δ is completed. Each item on this list is the generator matrix of a lattice having abscissa count N_{bsf} which, at this stage, coincides with N_{KO}. The CPU-time spent to construct this list is reported in the final column of the table.

Many of the lattices on the *best-so-far* list are identical lattices, specified by different generator matrices. $\nu(\delta)$ is the number of lattices remaining after the duplicates have been identified and discarded.

Again, many of these distinct lattices belong to the same equivalence class. $\mu(\delta)$ is the number remaining when all but one from any equivalence class is discarded. In fact, it was found that for all values of δ except one, all the lattices were in the same equivalence class. The exception occurred for $\delta = 8$, where lattices from two distinct equivalence classes were represented. The number of equivalence classes is given in the column headed $\mu(\delta)$.

The results given in Table 4 and illustrated in Figure 1 were obtained from the *best-so-far* list in a matter of minutes by using a post-processor to remove duplicates and retain only one in each equivalence class.

The overall impression from Table 3 is that of extensive redundancy in the computation. However, that is the tip of an iceberg. We have identified two major components of redundancy, in addition to the comparatively minor situation illustrated in the table.

The first component is a consequence of lattice duplication. It is not hard to see how this redundancy occurs. Dealing with a single quintet, the program examines all generator matrices that take as their rows one point on each of five different facet-pairs as specified by the quintet. An individual lattice may have several points on any particular facet. And in general a different generator matrix of the same lattice may be constructed using any choice of these points, as long as it uses only one from each facet pair. (The exceptions include the possibility of a singular matrix and the generation of a sublattice.) The program calculates individually the degree of all these different matrices, which in fact represent the identical lattice.

The second component arises from the separate treatment of many members of the same equivalence class. (The restriction from the 4,368 quintets to 11 has removed a gigantic number of symmetrically equivalent lattices but not those within the same quintet.) But each of the symmetrically equivalent lattices in the same quintet is treated separately; and, of course, each of these lattices is represented many times by different generator matrices.

Thus, in processing a single task, the client processes many generator matrices that have an identical abscissa count N_{bsf}. It retains and returns only one lattice, the first encountered having this common abscissa count.

The widescale redundancy mentioned above does not show itself in the table. The number $\nu_{bsf}(\delta)$ is the number that reach the *best-so-far* list, a very small proportion indeed of the number processed. However this list is itself highly redundant. For each value of δ, only one (or two in the case $\delta = 8$) independent lattice has been discovered.

We note that the description of the algorithm in Section 3.2 introduces the *best-so-far* list; the description there may leave the impression that this list, in the end, contains *all* the matrices that generate lattices having the optimal abscissa count N_{KO}. In fact, it contains only a small selection. This represents a compromise. If all had been retained, each client would have had to maintain its own *best-so-far* list, which would have to be transferred to the server. The subsequent communication, storage, and data-handling operations imposed at the server could easily have created a server bottleneck of the type described in Section 4, adding significantly to the overall CPU time. At the other extreme, if only one had been retained, there being effectively no *best-so-far* list but only a single *best-so-far* matrix, the postprocessing time might have been reduced by a matter of minutes, but the CPU time of the search itself would be essentially as reported.

In both cases, the correct abscissa count N_{KO} would be obtained, together with the generator matrix of a rule. (Naturally, all members of this equivalence class are also optimal rules.) The difference is as follows. In cases where there are optimal rules belonging to two or more equivalence classes (as in the $\delta = 8$ case) the lack of a *best-so-far* list means that the search is constrained to miss optimal rules belonging to all but one equivalence class.

Our compromise has several major advantages. The client need not retain a list; the information transfer between client and server is short. And the list retained by the server, while long, does not get out of hand. The final value of N_{KO} is properly obtained. The disadvantage is that the search *may* miss a whole equivalence class of optimal rules. (In retrospect, while conceding that this is possible, we deem it highly unlikely to have occurred in the search reported here.)

As mentioned at the end of Section 3.1, for $\delta > 8$ we restricted the search to a very small subset of $K(s, \delta)$. An appealing feature of the restricted search over $\mathcal{K}(s, \delta, q_3^+)$ is that the final component of the redundancy is significantly reduced. For $\delta = 7$ and 8, searches over both sets were carried out. Here we note that $T(\delta) \gg T(\delta, q_3^+)$ and $\nu_{bsf}(\delta) \gg \nu_{bsf}(\delta, q_3^+)$, while $\mu(\delta) = \mu(\delta, q_3^+)$. However, the reduced search found the same optimal lattices with a fraction of the work. We termed these rules K^+-optimal. We caution, however, that there is no guarantee that the restricted search has not completely missed all members of some equivalence class containing the K-optimal rule.

6. Results of the search

Our results are displayed in Table 4. These are nine five-dimensional lattice rules of enhanced degrees between 5 and 12. The first five are K-optimal rules (see Definition 2.2); the next four are K^+-optimal. The entries for $\delta = 12$ report the status of an incomplete search (approximately 3/4 of the search). At this point the search was terminated when the server was taken out of service.

The lattices are specified by the generator matrix of the dual lattice in Hermite Normal Form. Naturally only one lattice for each equivalence class is listed. Following convention this is the first one in a lexicographical order, called the *senior lattice* of this equivalence class (see [LySo93]).

A dimensionless measure for the quality of a lattice rule is provided by the *rho-index*, defined as

$$\rho(\Lambda) = \frac{\delta^s}{N \cdot s!}. \tag{15}$$

A thorough discussion of the rho-index is given in [Lyn03]. A lower bound on N provides an equivalent upper bound on $\rho(\Lambda)$. The bound corresponding to N_{ME} is denoted as ρ_{ME}. The best previously known set of five-dimensional rules was generated by circulant matrices [CoGo03], and these values are also shown in the table.

Figure 1 displays the rho-indices of all the rules in Table 4. This figure incorporates the information in the final three columns of Table 4 allowing an immediate visual comparison of rules.

7. Conclusions

The exhaustive search for five-dimensional K-optimal rules presented in this paper produced new rules better than previously known rule for enhanced degree 5 through 12.

This effort was made possible through a huge Internet-based search, which proved very efficient for this computation. However, even with the enormous computational power available to us through distributed Internet search, we reached a ceiling for what seems to be practically possible at $\delta = 8$ for the complete search and $\delta = 12$ for the restricted search.

When terminated, the program had consumed in total the equivalent of approximately 190 CPU-years spread across more than 1,400 computers during a four-year period. It is difficult to imagine this amount of computing power being awarded for this project by an individual supercomputing center.

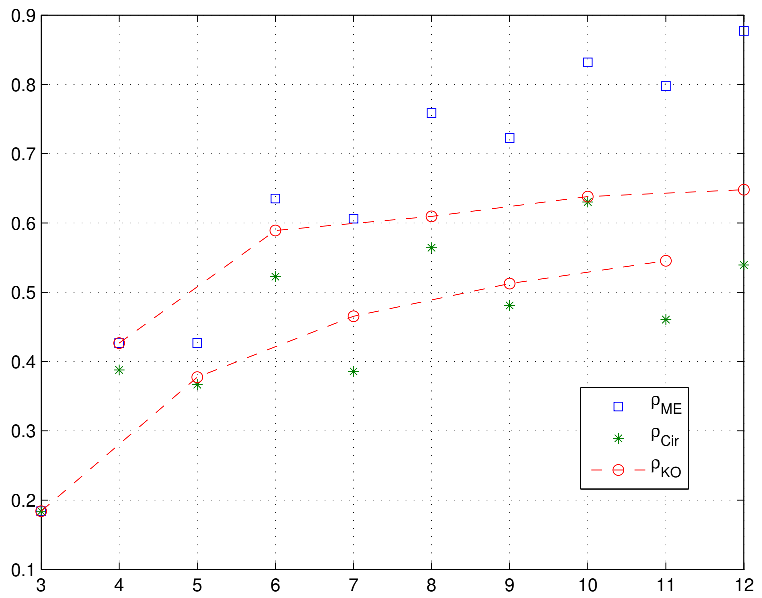

Figure 1. The rho-indices for three sets of rules. This information taken from the final three columns of Table 4. The two discontinuous lines connect ρ_{KO} for even values of δ and for odd values, respectively.

TABLE 4. Five-dimensional K-optimal lattice rules. Column 3 contains the rank of the rule and the number of distinct members of the equivalence class. Column 4 contains the abscissa count (and rho-index) of the K-optimal rule. Columns 5 and 6 contain corresponding information for hypothetical optimal rules based on the Möller-Mysovskikh bound (13) and for available optimal circulant rules [CoGo03].

δ	generator matrix	ν/ *rank*	N_{KO}/ρ_{KO}	N_{ME}/ρ_{ME}	N_{Cir}/ρ_{Cir}
5	1 0 0 0 4 0 1 0 0 13 0 0 1 0 19 0 0 0 1 29 0 0 0 0 69	1920 1	69 0.3774	61 0.4269	71 0.3668
6	1 0 0 0 15 0 1 0 0 21 0 0 1 0 25 0 0 0 1 33 0 0 0 0 110	1920 1	110 0.5891	102 0.6353	124 0.5226
7	1 0 0 0 6 0 1 0 0 45 0 0 1 0 61 0 0 0 1 81 0 0 0 0 301	1920 1	301 0.4653	231 0.6063	363 0.3858
8	1 0 0 0 9 0 1 0 0 61 0 0 1 0 101 0 0 0 1 157 0 0 0 0 448	1920 1	448 0.6095	360 0.7585	484 0.5642
8	1 0 0 0 15 0 1 0 0 35 0 0 1 1 24 0 0 0 4 104 0 0 0 0 112	480 2	448 0.6095	360 0.7585	484 0.5642
9	1 0 0 1 36 0 1 0 2 26 0 0 1 4 81 0 0 0 8 64 0 0 0 0 120	480 2	960 0.5126	681 0.7226	1023 0.4810
10	1 0 0 0 41 0 1 0 0 51 0 0 1 0 321 0 0 0 1 389 0 0 0 0 1306	1920 1	1306 0.6381	1002 0.8317	1322 0.6304
11	1 0 0 0 40 0 1 0 0 406 0 0 1 0 543 0 0 0 1 922 0 0 0 0 2461	1920 1	2461 0.5453	1683 0.7974	2913 0.4607
12	1 0 0 0 243 0 1 0 0 395 0 0 1 1 370 0 0 0 2 930 0 0 0 0 1600	1920 2	3200 0.6480	2364 0.8772	3844 0.5394

Acknowledgment: With gratitude we acknowledge the help of numerous, anonymous helpers who have donated their idle cycles to our computation. Without their help such comprehensive search could not have been accomplished. We are also in great debt to Jan Frode Myklebust, who has maintained the server and kept it running steadily for more than 4 years.

References

[CoGo03] R. Cools and H. Govaert *Five- and six-dimensional lattice rules generated by structured matrices*, Journal of Complexity **19** (2003), no. 6, 715–729

[CoLy01] R. Cools and J. N. Lyness *Three and four-dimensional K-optimal lattice rules of moderate trigonometric degree*, Math. Comp. **70** (2001), no. 236, 1549–1567.

[CoSl96] R. Cools and I. H. Sloan, *Minimal cubature formulae of trigonometric degree*, Math. Comp. **65** (1996), no. 216, 1583–1600.

[Lyn89] J. N. Lyness, *An introduction to lattice rules and their generator matrices*, IMA J. Numer. Anal. **9** (1989), 405–419.

[Lyn03] J. N. Lyness, *Notes on lattice rules*, Journal of Complexity **19** (2003), no. 3, 321–331.

[LyCo00] J. N. Lyness and R. Cools, *Notes on a search for optimal lattice rules*, in Cubature Formulae and Their Applications (M. V. Noskov, ed.), 259–273, Krasnoyarsk STU, 2000. Also available as Argonne National Laboratory Preprint ANL/MCS-P829-0600.

[LySø91] J. N. Lyness and T. Sørevik, *A search program for finding optimal integration lattices*, Computing **47** (1991), 103–120.

[LySø92] J. N. Lyness and T. Sørevik, *An algorithm for finding optimal integration lattices of composite order*, BIT **32** (1992), no. 4, 103–120.

[LySø93] J. N. Lyness and T. Sørevik, *Lattice rules by component scaling*, Math. Comp. **61** (1993), no. 204, 799–820.

[LySø04] J. N. Lyness and T. Sørevik, *Four-dimensional lattice rules generated by skew-circulant matrices* Math. Comp. **73** (2004), no. 245, 279–295.

[Min11] H. Minkowski, *Gesammelte Abhandlungen*, reprint (originally published in 2 volumes, Leipzig, 1911), Chelsea Publishing Company, 1967.

[Mol79] H. M. Möller, *Lower bounds for the number of nodes in cubature formulae*, in Numerische Integration (Tagung, Math. Forschungsinst., Oberwolfach 1978) Internat. Ser. Numer. Math. 45, Birhäuser, Basel 1979, 221–230.

[Mys88] I. P. Mysovskikh, *Cubature formulas that are exact for trigonometric polynomials*, Metody Vychisl. **15** (1988), 7–19 (Russian).

[Nos91] M. V. Noskov, *Cubature formulas for functions that are periodic with respect to some of the variables*, Zh. Vychisl. Mat. Mat. Fiz. **31** (1991), no. 9, 1414–1418 (Russian). Summary Math. Rev. 92i:65052.

[SøMy01] T. Sørevik and J. F. Myklebust, *GRISK: An Internet based search for K-optimal lattice rules*, in Proceedings of PARA2000, *Lecture Notes in Computer Science* 1947, 196–205, Springer Verlag, 2001.

Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439-4844, USA, and School of Mathematics, The University of New South Wales, Sydney 2052, Australia.

E-mail address: `lyness@mcs.anl.gov`

Department of Mathematics, University of Bergen, N-5020 Bergen, Norway.

E-mail address: `tor.sorevik@ii.uib.no`