A Two-Stage Approach for Computing Cubature Formulae for the Sphere

Jörg Fliege
Tel.: +49–231–755–5415
Fax.: +49–231–755–5416
Email: fliege@math.uni-dortmund.de

Ulrike Maier
Tel.: +49–231–755–3080
Email: umaier@math.uni-dortmund.de

Fachbereich Mathematik
Universität Dortmund
44221 Dortmund
Germany

In simulating interstellar dust clouds astrophysicists need high accuracy integration formulae for functions on the sphere. To construct better formulae than previously used, almost equidistantly spaced nodes on the sphere and weights belonging to these nodes are required. This problem is closely related to a facility dispersion problem on the sphere and to the theories of spherical designs and multivariate Gauss quadrature formulae.

We propose a two-stage algorithm to compute optimal facility locations on the unit sphere with high accuracy and an appropriate algorithm to calculate the corresponding weights of the cubature formulae. These algorithms can be extended to other facility dispersion problems. Numerical examples show that the constructed formulae yield impressive small integration errors of approximately 10^{-12}.

1 Introduction

In simulating interstellar dust clouds astrophysicists need high accuracy integration formulae for functions on the sphere. There are several known approaches to construct such cubature formulae. Aims are to obtain nodes inside the region of integration, to get positive weights to assure convergence results and to get a high degree of exactness.

Probably the best known approach are *product formulae* where univariate quadratures are composed to get multivariate formulae. Positive weights can be assured by using univariate Gauss formulae for this construction. For the sphere the main disadvantage is that lattice points lie much denser near the poles than in the equatorial region. In applications this point distribution is often not suitable.

Another approach are cubature formulae constructed with the aid of *invariant theory* (see e. g. Cools [6]) where the number of moment equations occuring in the calculation of nodes and weights are drastically reduced with the aid of *consistency conditions*. Here, classes of points with equal weights are used. The formulae in general do not have only positive weights and it cannot be decided in advance wether a formula of a certain degree exists or not.

In contrast to this, *ideal theory* makes it possible to decide the question of existence or nonexistence of a formula. Lower bounds for the number of nodes needed for a cubature formula of specific degree can also be obtained (see e. g. [6] and the references therein). Nodes are common zeros of certain orthogonal polynomials. However, the calculation of these zeros is problematic.

None of the formulae mentioned above is a *multivariate Gauss formula* which in the univariate case are known to be the formulae with highest degree of exactness and with general convergence due to their positive weights.

Of course there has also been some research concerning multivariate Gauss quadrature formulae (see e. g. Bos [3], Reimer [17, 18, 19]). It turned out that the existence of multivariate Gauss formulae for the sphere is closely related to the existence of cubature formulae with equal weights. Multivariate

Gauss formulae for the sphere can even be characterized by this property [17]. Bos [3] has proved that nodes yielding equal weights also solve the Fejér problem, that is the square-sum of the Lagrange polynomials is equal to one for these nodes. This property leads to results concerning the existence or nonexistence of multivariate Gauss quadrature formulae [3].

Nodal systems yielding cubature formulae with equal weights on the sphere are also known as *spherical designs*. Here, the number of nodes is not restricted to the dimension of the corresponding polynomial space. There are several results on the existence and characterization of spherical designs (Delsarte, Goethals and Seidel [8], Seidel [21], Bannai and Damerell [1, 2]), but no algorithm for the calculation of these designs is known.

An estimation of the integration error for equally weighted cubature formulae was recently given by Cui and Freeden [7]. However, convergence for this approach has up to now only be proved for spherical designs.

After having finished our numerical calculations we discovered that Zhou [25] has partially worked on the same subject. For his thesis he minimized numerically several objective functions using quasi-Newton methods. Zhou puts an emphasis on geometric considerations but does not inspect cubature formulae. He restricts his computations to facility numbers ≤ 200.

Steinacker, Thamm and Maier [23] computed quadrature formulae for the sphere, yielding better results then the approach of Reimer and Sündermann [20], but the algorithm for the node distribution is not an optimal one.

In this paper the terms facilities, nodes and points are used for the same object without further notion.

We propose a two-stage algorithm using first a stochastical algorithm and an improved quasi-Newton method to compute nearly equidistant nodes on the sphere and then to calculate corresponding weights which differ only little from equal weights. Nodes as well as weights are computed with high precision and the resulting cubature formulae are tested for several functions. We computed point systems for up to 1600 nodes and calculated cubature weights for up to 900 points.

This article is organized as follows. Section 2 summarizes results about spherical designs and multivariate Gauss quadrature formulae. In Section 3 our technique for computing nodes and weights is described. Section 4 contains implementation details and Section 5 presents numerical results.

For $r \in \mathbb{N}$ the unit sphere in $\mathbb{R}^r$ is denoted by S^{r-1}. For $m \in \mathbb{N}_0$ let $\mathbb{P}_m^r(S^{r-1})$ denote the linear space of polynomial restrictions on S^{r-1} in r variables of degree $\leq m$ and let $\overset{*}{\mathbb{P}}{}_m^r\,(S^{r-1})$ denote the corresponding space of homogeneous polynomials.

A system $X = \{x_1, \ldots, x_N\}$ of nodes $x_1, \ldots, x_N$ is called a *fundamental system* with respect to $\mathbb{P}_m^r(S^{r-1})$ ($\overset{*}{\mathbb{P}}{}_m^r\,(S^{r-1})$), if the corresponding evaluation functionals form a basis in its dual space. This property ensures that the corresponding *Lagrange fundamental polynomials* l_j are well defined and that $l_j(x_k) = \delta_{jk}$ holds (δ_{jk} the Kronecker symbol).

Further let $\langle\, f, g \,\rangle := \int\limits_{S^{r-1}} f(x)g(x)dx$ be the surface integral of S^{r-1}. The reproducing kernel of $\mathbb{P}_m^r(S^{r-1})$ is denoted by G_m and has the representation

$$G_m(x, y) = \frac{1}{\mu(S^{r-1})} \left[C_m^{r/2}(x^T y) + C_{m-1}^{r/2}(x^T y) \right], \tag{1}$$

where $\mu(S^{r-1}) = \langle\, 1, 1 \,\rangle$ is the surface area of S^{r-1} and $C_m^{r/2}$ denotes the *Gegenbauer polynomial* of degree m with index $r/2$. The reproducing kernel of $\overset{*}{\mathbb{P}}{}_m^r\,(S^{r-1})$ is denoted by $\overset{*}{G}_m$ and has the representation

$$\overset{*}{G}_m\,(x, y) = \frac{1}{\mu(S^{r-1})} C_m^{r/2}(x^T y). \tag{2}$$

2 Existence of Cubature Formulae with Equal Weights

Even in the univariate case there are only a few quadrature formulae with equal weights. These are quadrature formulae for the weighted integral with the weight function of the Tschebyscheff-polynomials of the first kind or for low degrees formulae for integrals with a constant weight function. More exactly we have the following theorems (see e. g. Krylov [13]).

Theorem 1 *If for arbitrary values of $N = 1, 2, \ldots$ there exist constants c_N such that*

$$\int_{-1}^{1} w(x) f(x) dx = c_N \sum_{k=1}^{N} f(x_k)$$

holds for $f(x) = 1$, $f(x) = x$, $f(x) = x^2$, then $w(x)$ coincides with the Tschebyscheff weight function $(1 - x^2)^{-1/2}$.

Theorem 2 *For $N \geq 10$ there is no formula*

$$\int_{-1}^{1} f(x) dx \approx \frac{2}{N} \sum_{k=1}^{N} f(x_k)$$

with only real x_k, $k = 1, \ldots, N$, that is exact for all polynomials of degree $\leq N$.

The univariate results show that the existence of quadrature formulae with equal weights cannot be expected too often in the far more complicated multivariate case. Results of the theory of spherical designs and multivariate Gauss quadrature formulae confirm this conjecture [1, 2, 3].

2.1 Spherical Designs

In this subsection we give a brief overview of the most important results concerning spherical designs.

Definition 1 *([8]) Let $X \subset S^{r-1}$ and $A(X) := \{\langle \xi, \eta \rangle, \xi \neq \eta \in X\} \subset [-1, 1]$. Then X is called a* spherical A-code. *A finite subset $X \subset S^{r-1}$ is a* spherical design *of strength t (for short t-*design*) if*

$$\frac{1}{|X|} \sum_{x \in X} f(x) = \frac{1}{\mu(S^{r-1})} \int\limits_{S^{r-1}} f(\xi) d\mu(\xi) \quad \textit{for every} \quad f \in I\!P_t^r(S^{r-1}).$$

Here $|X|$ denotes the number of elements of X and μ the surface measure of the sphere.

Theorem 3 *([8]) Let X be a $(2m)$-design. Then $|X| \geq G_m(1)$. Equality holds if and only if $A(X)$ consists of the zeros of $G_m(x)$.*

Theorem 4 *([8]) Let X be a (2m+1)-design. Then $|X| \geq 2 \overset{*}{G}_m (1)$. Equality holds if and only if $A(X)$ consists of -1 and the zeros of $\overset{*}{G}_m (x)$. Moreover, in the case of equality the design X is antipodal (i.e. $X = -X$).*

Definition 2 *([8]) A t-design is called* tight *if any of the bounds is attained.*

Remark 1

$$\begin{aligned} G_m(1) &= \binom{m+r-1}{r-1} + \binom{m+r-2}{r-1} = \dim I\!P^r_m(S^{r-1}), \\ \overset{*}{G}_m (1) &= \binom{m+r-1}{r-1} = \dim \overset{*}{I\!P}{}^r_m (S^{r-1}). \end{aligned}$$

A necessary and sufficient condition for t-designs that uses special matrix equations is given in [8].

Example 1 *([8])*

(i) For $r = 2$ and any t, a tight t-design is the set of all vertices of a regular $(t+1)$-gon.

(ii) For $r = 3$ the icosahedron is the only tight 5-design.

(iii) For any r, the $r+1$ vertices of a regular simplex in $\mathbb{R}^r$ provide a tight 2-design.

The following two theorems show that tight spherical designs and thus equally weighted quadrature formulae scarcely exist for $r \geq 3$.

Theorem 5 *([1]) Let $t = 2m$, $m \geq 3$ and $r \geq 3$. Then there exists no tight spherical t-design in S^{r-1}.*

Theorem 6 *([2]) Let $t = 2m+1$, $m \geq 4$ and $r \geq 3$. Then there exists no tight spherical t-design in S^{r-1} except for $t = 11$ and $r = 24$.*

2.2 Gauss Quadrature Formulae

The following theorem links the Fejér-problem directly with the existence problem of multivariate Gauss-quadratures and in case of $\mathbb{P}^r_m(S^{r-1})$ with the existence of a $2m$-design.

Theorem 7 *([3]) Suppose that $x_1, \ldots, x_N \in S^{r-1}$ is a fundamental system. Then $\max\limits_{x \in S^{r-1}} \sum\limits_{j=1}^{N} l_j^2(x) = 1$ if and only if $\frac{1}{\mu(S^{r-1})} \int\limits_{S^{r-1}} p(x)dx = \frac{1}{N}\sum\limits_{j=1}^{N} p(x_j)$ for all $p \in \mathbb{P}^r_{2m}(S^{r-1})$.*

Theorem 8 *([3]) Suppose that $r \geq 3$ and $m \geq 3$ and $x_1, \ldots, x_N \in S^{r-1}$ maximizes $VDM(x_1, \ldots, x_N)$ over S^{r-1}, where VDM is the Vandermonde determinant. Then $\max\limits_{x \in S^{r-1}} \sum\limits_{j=1}^{N} l_j^2(x) > 1$.*

Thus, for $r \geq 3$, $m \geq 3$ there exist no Gauss-quadratures for $\mathbb{P}^r_m(S^{r-1})$ and for $r \geq 3$, $m \geq 4$ they do not exist for $\overset{*}{\mathbb{P}}{}^r_m(S^{r-1})$. There might be Gauss-quadratures for $m = 2$. But even in these cases it turns out that certain restrictions have to hold.

Theorem 9 *([17]) If a Gauss-quadrature exists on $\mathbb{P}^r_2(S^{r-1})$, $r > 2$, then r has the form $r = p^2 - 3$ where $p \in \mathbb{N} \setminus \{1\}$ is odd. Hence the first dimensions where existence is possible are given by $r \in \{2, 6, 22, 46, 78, 118\}$. Existence is known for $r \in \{2, 6, 22\}$.*

Theorem 10 *([17]) For $r \geq 3$ a Gauss-quadrature rule on $\overset{*}{\mathbb{P}}{}^r_2(S^{r-1})$ exists if and only if the same is true for $\mathbb{P}^{r-1}_2(S^{r-2})$.*

That is, existence is possible for $r \in \{2, 3, 7, 23, 47, 79, 119\}$ and it is known for $r \in \{2, 3, 7, 23\}$.

3 A Two-Stage Approach

3.1 The First Stage: Finding the Points

In the univariate case the zeros of Jacobi-polynomials give Gauss-Jacobi quadratures which are distinguished by highest possible degree of exactness.

Moreover, these quadratures have positive weights from which convergence of these formulae is guaranteed.

To advance to the multivariate case, an interpretation of the zeros of the Jacobi polynomials is helpful. The zeros of these orthogonal polynomials are closely connected to movable charges situated on a metal rod which is represented by the real interval $[-1,1]$. In the state of equilibrium the potential energy of the charge distribution is minimized and their locations coincide with the zeros of the Jacobi polynomials. For more details see Stroud and Secrest [24, p. 17].

Thus a minimization of the potential energy of a point distribution on the sphere is a good criterion for an appropriate node distribution.

With $\|\cdot\|$ the Euclidian norm in $\mathbb{R}^r$ we get for electrical charged point particles at locations $x_1,\ldots,x_N \in S^{r-1}$ the potential energy

$$E(x_1,\ldots,x_N) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{1}{\|x_i - x_j\|} \tag{3}$$

and thus the nonlinear optimization problem

$$\begin{aligned} &\text{minimize} \qquad E(x_1,\ldots,x_N) && (4)\\ &\text{subject to} \qquad x_i \in S^{r-1} \qquad (1 \le i \le N), && (5) \end{aligned}$$

which is a *facility dispersion problem* on the sphere. In computational physics and chemistry, this problem is also known as *Thomson's problem.*

3.2 The Second Stage: Finding the Weights

In the cases where the calculated nodes can be shown to be spherical designs ($N = 2,4,6,12$), equal weights are optimal. This can be seen with the aid of Theorems 3 and 4. In all other cases appropriate weights have to be calculated.

With the aid of multivariate interpolation theory this is at least possible if the number N of nodes equals the dimension of the linear space of polynomials $\mathbb{P}^r_m(S^{r-1})$. For $r = 3$ this means that $N = (m+1)^2$.

Let $x_1, x_2, \ldots, x_N \in S^{r-1}$ and let $G := (G_m(x_j, x_k))_{j,k=1}^N$ be the symmetric matrix being composed of the evaluation of the kernel G_m at the nodes x_i, $i = 1,\ldots,N$.

If $x_1, x_2, \ldots, x_N$ are a fundamental system, i. e. $\det(G) \neq 0$, then the interpolation property

$$G_m(x_j, x) = \sum_{k=1}^{N} G_m(x_j, x_k) l_k(x), \quad j = 1, \ldots, N \tag{6}$$

and the following consequence of the reproducing property of G_m

$$l_j(x) = \int\limits_{S^{r-1}} l_j(y) \cdot \left(\sum_{k=1}^{N} G_m(x_k, x) l_k(y) \right) dy \tag{7}$$

are valid. Here, l_j denote the Lagrange fundamental polynomials. Equations (6) and (7) lead to

$$\sum_{k=1}^{N} \left(\int_{S^{r-1}} l_j(y) l_k(y) dy \right) \cdot G_m(x_k, x_i) = \delta_{ji}, \tag{8}$$

which means that the matrices $L := (\langle\, l_j, l_k \,\rangle)_{j,k=1}^{N}$ and $G = (G_m(x_j, x_k))_{j,k=1}^{N}$ are inverse to each other.

Consider now a cubature formula

$$\int\limits_{S^{r-1}} f(x) dx = \sum_{j=1}^{N} \omega_j f(x_j), \quad f \in C(S^{r-1}). \tag{9}$$

The function f can be substituted in the usual way by its interpolation polynomial p with regard to the nodes $x_1, x_2, \ldots, x_N$. Then

$$\int\limits_{S^{r-1}} p(x) dx = \sum_{j=1}^{N} \left(\int_{S^{r-1}} l_j(x) dx \right) f(x_j),$$

that is

$$\omega_j = \int\limits_{S^{r-1}} l_j(x) dx.$$

Because $1 \in I\!P_m^r(S^{r-1})$ and $\sum_{k=1}^{N} l_k(x) = 1$ this yields

$$\omega_j = \sum_{k=1}^{N} \left(\int_{S^{r-1}} l_j(x) l_k(x) dx \right) \tag{10}$$

which is the row-sum of the matrix L.

Thus, because of (1), the weights of the cubature formula can be computed as the row-sum of the symmetric matrix

$$L = G^{-1} = \left[C_m^{r/2}(x_j^T x_k) + C_{m-1}^{r/2}(x_j^T x_k) \right]_{j,k}^{-1}. \tag{11}$$

The Gegenbauer polynomials of degree m with index $r/2$ occuring in this composition have the representation

$$C_m^{r/2}(\xi) \quad = \quad \sum_{k=0}^{[m/2]} (-1)^k \frac{\frac{r}{2}(\frac{r}{2}+1)\cdots(\frac{r}{2}+m-k-1)}{(m-2k)!\,k!}(2\xi)^{m-2k}$$

with $\xi \in [-1,1]$.

They can be evaluated with the aid of their recurrence relation (see e. g. Reimer [16])

$$(m+1)C_{m+1}^{r/2}(\xi) - 2\left(m+\frac{r}{2}\right)\xi C_m^{r/2}(\xi) + (m+r-1)C_{m-1}^{r/2}(\xi) = 0. \quad (12)$$

The inversion of the matrix $G = (G_m(x_j,x_k))_{j,k=1}^N$ can be avoided. With

$$e := (1,\ldots,1)^\top \in \mathrm{IR}^N$$

we obtain the vector $\omega := (\omega_1,\ldots,\omega_N)^\top$ of weights as solution of the linear system of equations

$$G\omega = e. \quad (13)$$

In the following, we study the case $r = 3$, that is $N = (m+1)^2$, in detail, but the theory can be extended to higher dimensions without any difficulties.

The cases $N \neq (m+1)^2$ cannot be handled yet except for those cases in which the nodes are spherical designs. These are the cases $N = 2,4,6,12$. The weights then can be chosen to be $4\pi/N = \mu(S^2)/N$. The cases $N = 3,5,7$ also show a regular behaviour, but up to now it could not be decided if these nodes are spherical designs. As interpolatory formulae our cubatures have a degree of exactness equal to m, whereas the spherical designs yield a degree of exactness equal to $2m$ or $2m+1$, respectively.

4 Implementation Details

In this section we describe details about our implementation for computing integration formulae for the case $r = 3$. These programs can easily be extended such that cubature formulae for higher dimensions are computed.

4.1 Reformulating the Objective Function

The nonlinear optimization problem (4)–(5) has as constraints the relations $x_i \in S^{r-1}$ $(1 \leq i \leq N)$, which can be written as $\|x_i\|_2 = 1$. However, these nonlinear equality constraints are computationally not so easy to handle as a corresponding reformulation involving spherical coordinates. With these, we can write every point $x_i \in S^2 = S^{r-1}$ as

$$S^2 \ni x_i = x(\varphi_i, \psi_i, r_i) = \begin{pmatrix} r_i \cos(\varphi_i) \sin(\psi_i) \\ r_i \sin(\varphi_i) \sin(\psi_i) \\ r_i \cos(\psi_i) \end{pmatrix}.$$

Of course, it is $r_i = 1$. Therefore, each point x_i on the three-dimensional unit sphere can be parametrized by only two variables $\varphi_i \in [0, 2\pi]$ and $\psi_i \in [0, \pi]$. Our problem now becomes

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{N} \sum_{j=i+1}^{N} d(\varphi_i, \psi_i, \varphi_j, \psi_j) && (14) \\ &\text{subject to} && 0 \leq \varphi_i \leq 2\pi \qquad (1 \leq i \leq N) && (15) \\ & && 0 \leq \psi_i \leq \pi \qquad (1 \leq i \leq N), && (16) \end{aligned}$$

where

$$\begin{aligned} d(\varphi_i, \psi_i, \varphi_j, \psi_j) \quad := \quad & \Big((\cos(\varphi_i) \sin(\psi_i) - \cos(\varphi_j) \sin(\psi_j))^2 \\ & + (\sin(\varphi_i) \sin(\psi_i) - \sin(\varphi_j) \sin(\psi_j))^2 \\ & + (\cos(\psi_i) - \cos(\psi_j))^2 \Big)^{-1/2}. \end{aligned}$$

The actual number of unknowns has dropped from $3N$ to $2N$. Moreover, we have traded the N quadratic equality constraints $\|x_i\|_2^2 - 1 = 0$ against simple box constraints in IR^{2N}. On the other hand, the objective function involves now trigonometric terms, which results in a subsequent higher computational effort for an objective function evaluation. Nevertheless, our computational tests with both formulations clearly indicate that the reformulation is favorable. We may also assume without loss of generality that the facility x_1 is fixed at the north pole,

$$x_1 := (0, 0, 1)^\top, \qquad \varphi_1 := \pi/4, \quad \psi_1 := \pi/2.$$

Moreover, the longitude of x_2 can also be fixed:

$$\varphi_2 := 0.$$

The N-facility problem has now $2N - 3$ unknowns. Without fixing x_1 at the north pole and simultaneously fixing the longitude of x_2, every local minimum can be transformed into another one by an arbitrary rotation of the facility locations. But even now will arbitrary permutations between the facility locations produce different optima.

4.2 Computing the Facilities

We have to solve the difficult nonlinear nonconvex constrained global optimization problem (14)–(16), which has a high number of local minima. In order to solve this problem according to our accuracy demands, we propose a two-phase approach. In Phase I, only a rough approximation of a global optimum is computed. This is done with a stochastic optimization method. In Phase II, the previously computed approximation is refined with a highly accurate nonlinear local optimization method. It is well known (Ingber [12]) that stochastic optimization methods approach very rapidly a global optimum, but converge very slowly once they are in a certain neighbourhood of an optimum. The shunt to another, locally faster method should therefore result in an overall better performance. Our computational results suggests that this is true, at least for the objective function at hand. This is also confirmed by results of Steinacker, Thamm and Maier [23], who only used a stochastic method to compute a highly accurate approximation to a global optimum. A short survey of other methods which have been proposed to attack this problem is given in [14].

We also intend to show that there exist optimization codes in the public domain which are comparable in performance with commercial ones. To this end, we decided to use only careful selected free implementations for our computational study.

4.2.1 Phase I

We choose a standard simulated annealing approach for continuous global optimization problems to compute approximations of global optima. Since there are already good implementations of several codes in the public domain, we choose the code of Goffe, Ferrier and Rogers [11] as a starting point for our own implementation. After preliminary testing, we found that their code outperformed the code of Ingber [12] when applied to the facility dispersion problem above. However, we have not tried to fine tune every possible parameter in the implementation of Ingber. Doing this may result in significant performance gains.

It turned out that after the implementation of the objective function some changes in the parameters were necessary in order to have a program that produces good results. We list here only the most important parameters, for details about the implementation we refer to [10, 11].

We set the starting temperature of the simulated annealing algorithm to $T := 5.0$. If N denotes the total number of facilities in the problem, the temperature is reduced after $5N$ steps according to $T_{\text{new}} := T_{\text{old}}/4$. The maximum number of function evaluations is set to $100N$. The algorithm stops when the maximum number of function evaluations is achieved or when the changes in the best known function values during the last four temperature reductions and the differences between the function values in the last step are less than $\varepsilon := (N+1)(N+2)/4000$. Note that, due to the peculiar chosen parameters, at most 20 temperature reductions take place. Since the temperature is reduced by a fixed amount, the algorithm used is in fact a specific form of simulated quenching, not simulated annealing. However, after experimenting with different annealing schemes, we never found that the algorithm got stuck in a nonglobal minimum, even with such a strict annealing schedule.

As it can be seen, the stopping criterion based on function values is not a particular strict one. This is the case because there is no necessity for high accuracy in Phase I. The algorithm of the second phase will converge to the minimum approached by the simulated annealing algorithm much faster than a stochastic method.

4.2.2 Phase II

To refine the facility distribution computed in Phase I, we used a limited memory BFGS-method (L-BFGS) implemented by Zhu [27]. We also experimented with a specific implementation of a quasi-Newton method [9] and a truncated Newton method [15], but our computational experience suggests that the L-BFGS code is faster and more reliable on our problem than the two other programs. Since details about the implementation used can be found elsewhere [27, 4, 5], we discuss here only changes introduced in order to maximize the performance of the code on our specific problem. Here, f^k and f^{k-1} are the function values from the current and the last step, $\mathrm{proj}(\nabla f)$ is the gradient at the current point projected on the $(2N-3)$-dimensional cube defined by the constraints and `eps` is the machine precision.

To compute a solution of the facility dispersion problem with highest possible accuracy, we implemented an exact gradient evaluation and used the following stopping criterion. The code stops when

$$1000(f^{k-1} - f^k) \leq \mathtt{eps} \cdot \max\{f^k, f^{k-1}, 1\} \tag{17}$$

or when

$$10000 \parallel \mathrm{proj}(\nabla f) \parallel_\infty < 1 + f^k \tag{18}$$

or when the line search was unsuccessful. (The line search consists of successive polynomial interpolation and tries to enforce certain curvature conditions. A line search is deemed to be unsuccessful if there are more than 10 function evaluations during the search.) We decided to use the first stopping criterion (17) because of our high accuracy demand. This criterion is an extremely conservative one in the sense that rounding errors will usually prevent the inequality from becoming true.

Our computational results suggest that only one run of the simulated annealing algorithm, followed by one run of the L-BFGS method is necessary to produce a good approximation to a global optimum. Moreover, weaker stopping criteria than the ones described above usually led to integration formulae which do not perform as well on test functions as formulae computed with stricter termination rules.

4.3 Computing the Weights

To secure a numerically stable computation of the matrix G, the Gegenbauer polynomials are evaluated with the aid of their recurrence relation (12).

For each of the $N(N+1)/2$ matrix entries which have to be computed we need 3 arithmetic operations for the scalar products of the vectors x_j and x_k and $4\sqrt{N}$ arithmetic operations for the evaluation of the Gegenbauer-polynomials using the recurrence relation (12). Thus, the computation of G requires $O(N^{5/2})$ arithmetic operations. Because of the symmetry of G we used Cholesky-decomposition to solve the system of equations (13). This requires $N^3/6 + 3N^2/2 + N/3$ operations. The calculation of the weights therefore needs $O(N^3)$ arithmetic operations.

5 Computational Results

5.1 The Facilities

The optimization codes were compiled with the `f77` compiler using the `-O3` (aggressive code optimization) compilation flag on a Silicon Graphics Power Indigo^2 with Mips R8000 chip, 192 MB main memory and IRIX 6.0.1 operating system. The machine precision on this machine is approximately 2.22×10^{-16}. All facility distributions were computed on this machine. The computation times were obtained by the IRIX `time` command.

We computed facility distributions for problems with N facilities, $N = (m+1)^2$, $m = 1, \ldots, 39$, and, additionally, $N = 1, \ldots, 200$. The last set of distributions was computed for benchmarking reasons. The biggest problem solved had therefore $N = 1600$ facilities, which resulted in a nonlinear nonconvex global optimization problem with 3197 unknowns and 6394 constraints.

Figure 1 shows the computation time for the first (simulated annealing, SA) and the second (limited-memory BFGS, L-BFGS) phase in Stage 1. A detailed table of all computation times can be found in the appendix, see Table 2. As it can be seen, the bottle neck of the computation is the first part of the optimization process, i. e. obtaining a point distribution which is not too bad. Note that only one run of the simulated annealing algorithm in Phase I was needed to produce a good starting distribution for Phase II. The biggest problem took 722 minutes (about 12 hours) computation time in Phase I, followed by another 79 minutes for Phase II.

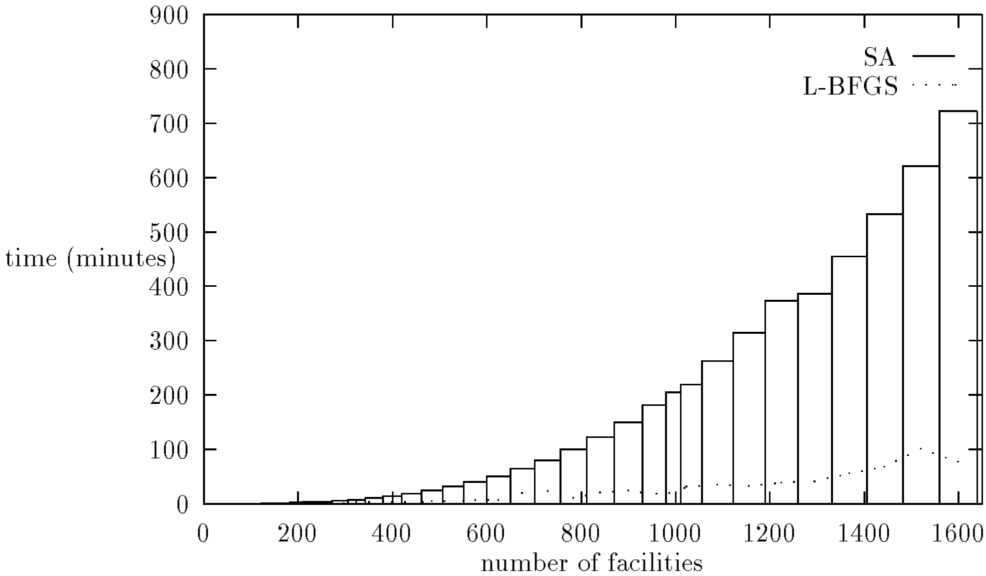

Figure 1: Time needed in Stage 1

Figure 2 shows that the quasi-Newton method employed by us produces a facility distribution very close to an optimal one. On a typical run, we had $\| \operatorname{proj}(\nabla f) \|_\infty \approx 100$ for the distribution calculated by the simulated annealing algorithm. The L-BFGS algorithm then refined this number to $\| \operatorname{proj}(\nabla f) \|_\infty \approx 0.0001$. Detailed results can be found in Table 3 in the appendix.

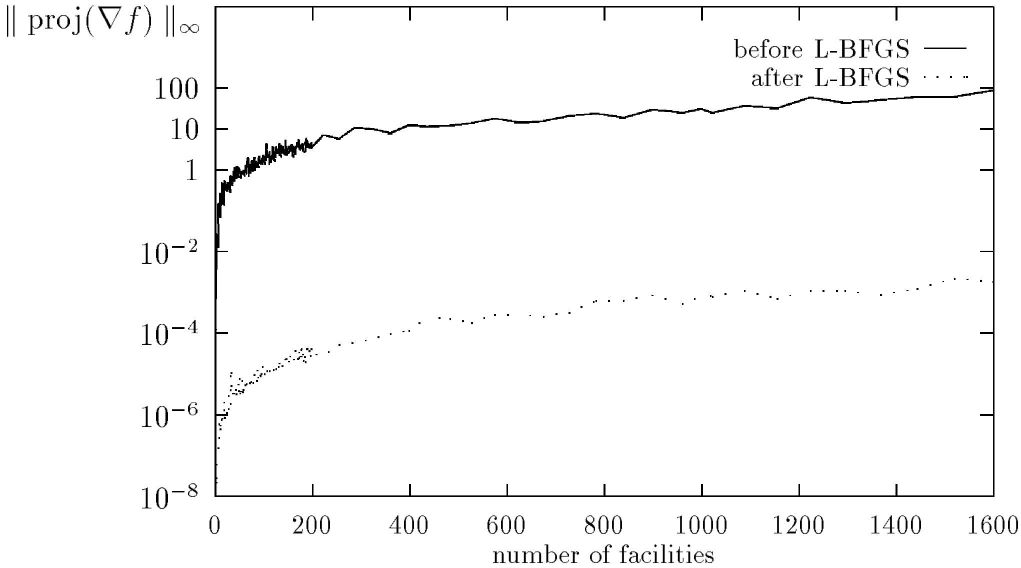

Figure 2: Norm of the projected gradient before and after Phase II

Tables 4 and 5 (see Appendix) confirm that the optimal function values found by our approach are at least as good as the function values reported in [25, 14, 23].

Figures 3–6 show the computed facility distributions for different numbers of nodes.

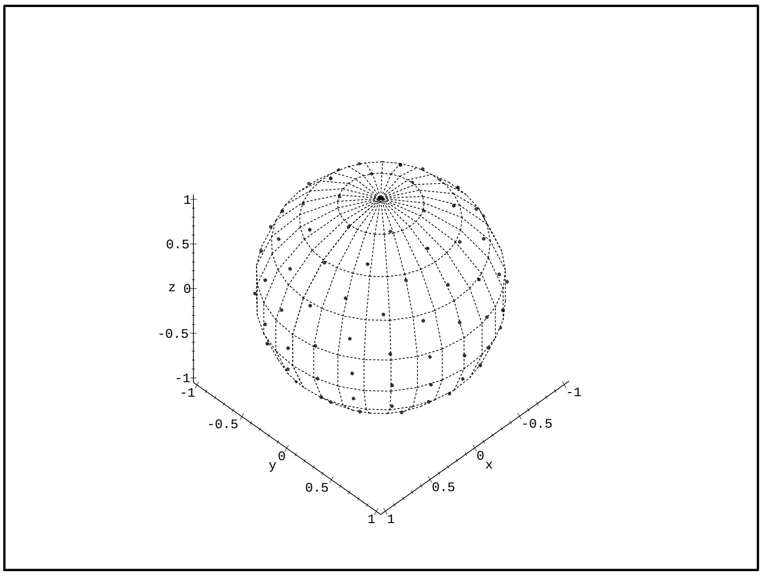

Figure 3: 121 nodes on the sphere

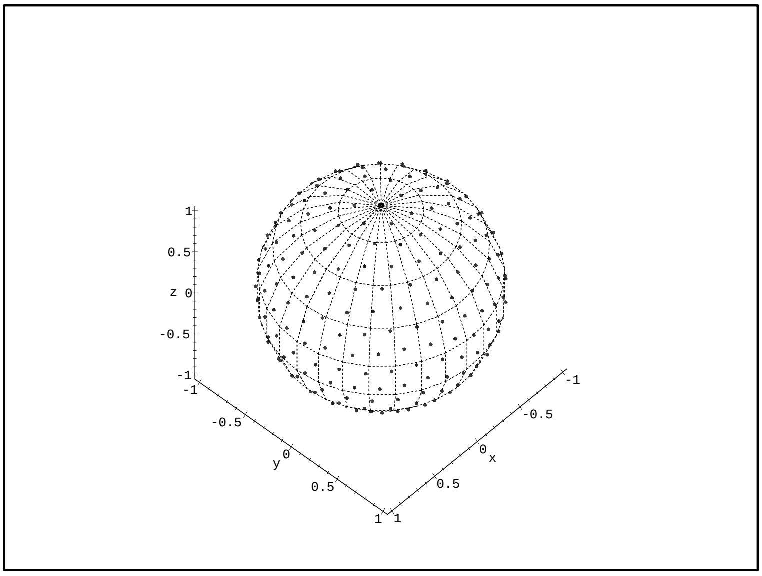

Figure 4: 324 nodes on the sphere

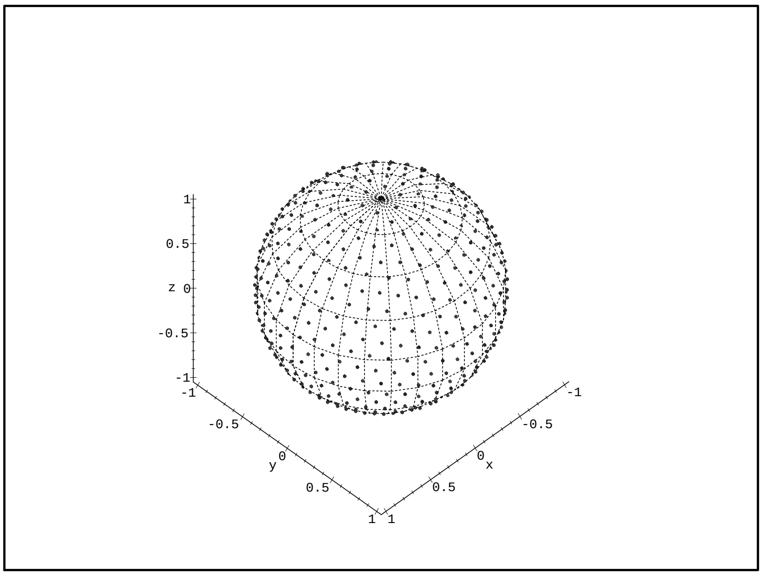

Figure 5: 625 nodes on the sphere

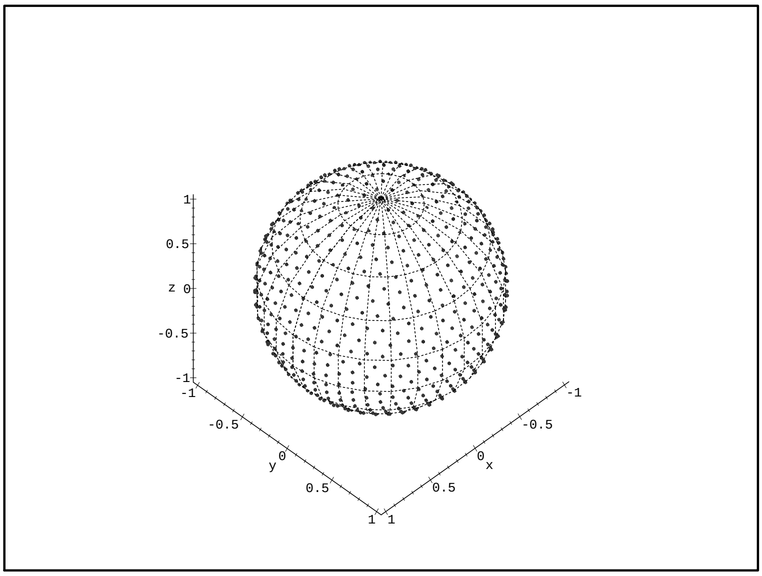

Figure 6: 900 nodes on the sphere

It has to be noted that the nodes calculated by us are no extremal fundamental systems in the sense of Reimer [16]. Using our nodes as starting points for the algorithm of Reimer and Sündermann [20] we found that our points had $\|l_j\|_\infty \approx 1$, $j = 1, \ldots, N$, but their algorithm nevertheless produced completely different facility distributions.

5.2 The Weights

The code for computing the weights was developed with SPARCompiler Pascal 3.0 and cross-compiled to C with `p2c`, version 1.20. The actual compilation took place with the `cc` compiler (again using the `-O3` flag) on a SPARCstation 20 with 64 MB main memory and SunOS 5.3 operating system.

On this machine we computed the optimal weights for the facility distributions from the first stage with $N = (m+1)^2$ facilities ($m = 1, \ldots, 29$) (see

Section 3.2).

Computation times were obtained by the UNIX `time` command. Figure 7 shows these computation times for Stage 2.

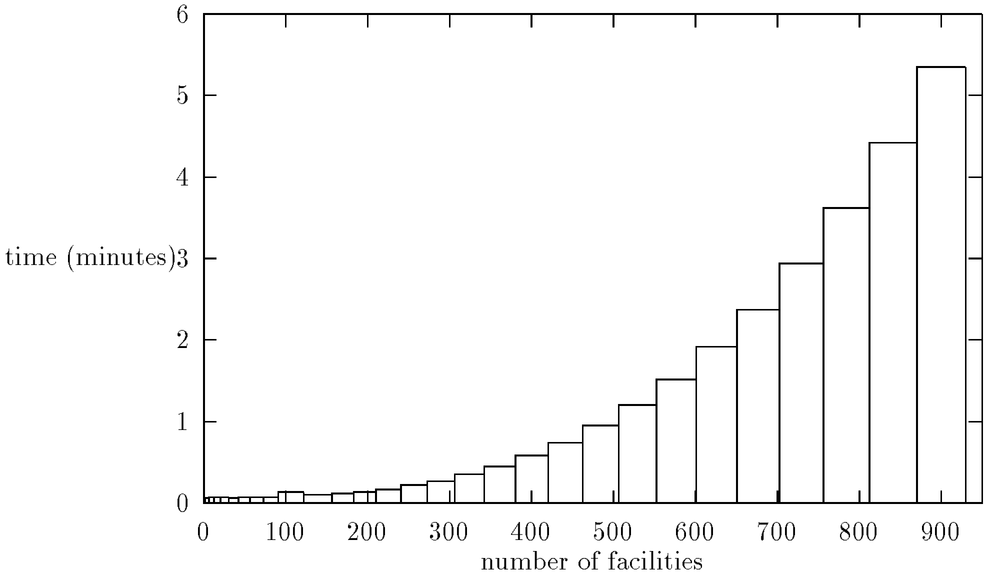

Figure 7: Time needed in Stage 2

The graphic shows clearly that the difference between the two computing environments for computing the facility locations and computing the weights played no role whatsoever in the computational process, since the resources needed to compute the weights are far smaller than the resources needed to compute the locations of the facilities.

Concerning the distribution of weights, our weights show a deviation of about $1.1 \cdot 10^{-2} - 8.9 \cdot 10^{-2}$ for $N = 16, 25, 36, 49$ and $4.1 \cdot 10^{-3} - 1.1 \cdot 10^{-2}$ for $N = 64, 81, 100$, whereas the extremal fundamental systems of Reimer and Sündermann [20] differ from equal weights by about $3.2 \cdot 10^{-2} - 1.4 \cdot 10^{-1}$ for $N = 16, 25, 36, 49$. Moreover, our weights are closer to equal weights than the weights from [23]. All of our weights are positive except for some isolated cases.

5.3 Testing the Formulae

We tested our computed integration formulae on six different functions described in Table 1.

j	$f_j(x)$	$\int_{x\in S} f_j(x)dx$
1	$\exp(x_1+x_2+x_3)/10$	$1.98622365460000\ldots$
2	$\Vert x \Vert_1 /10$	$1.88495559215388\ldots$
3	$-5\sin(1+10x_3)$	$2.87630387748651\ldots$
4	$1/(10.1-10x_3)$	$3.33216474778102\ldots$
5	$\exp(x_1)$	$14.7680137457653\ldots$
6	$x_1x_2x_3$	0

Table 1: Test functions and exact integration values

The absolute integration error for an N-node formula is now defined as

$$E_N(j) := \left| \int_{S^2} f_j(x)dx - \sum_{i=1}^{N} \omega_i f_j(x_i) \right|, \quad j = 1, \ldots, 6.$$

Figure 8 and 9 show these integration errors for all computed formulae and all test functions. A detailed list of integration errors is contained in Tables 6 and 7 (see appendix). Since the machine precision is about 10^{-16}, the computed formulae have a "treshold" at about 10^{-12}. Without a doubt, the errors will get smaller when the numerical integration is done on a machine with higher machine precision (or which simulates more accurate floating point numbers by software).

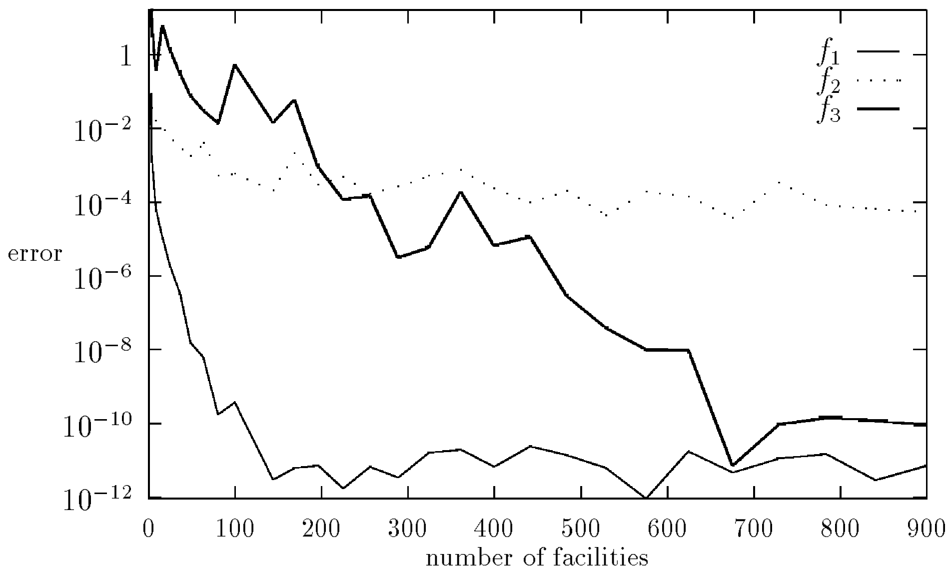

Figure 8: Absolute error for test functions f_1, f_2 and f_3

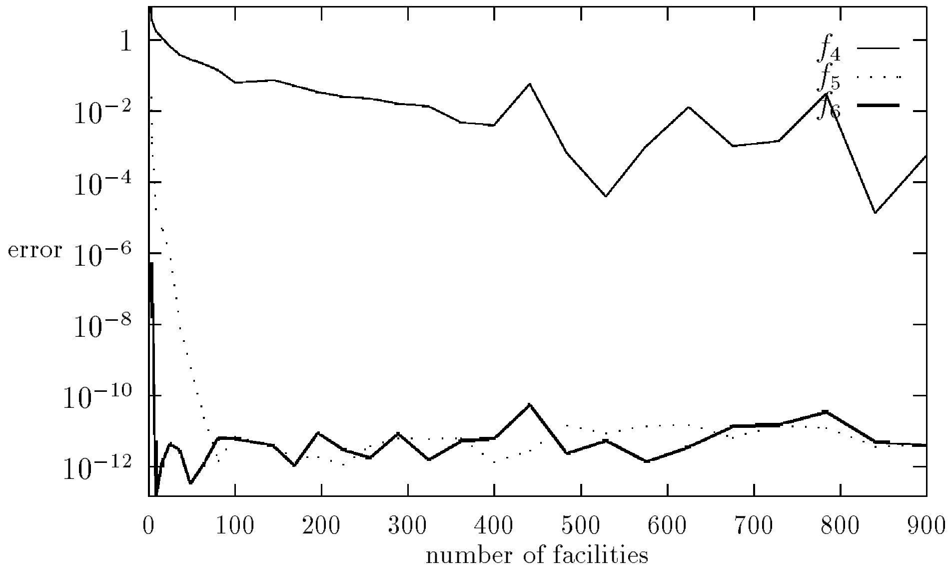

Figure 9: Absolute error for test functions f_4, f_5 and f_6

Unfortunately, to the best of our knowledge, comparable numerical results for integration formulae do not seem to be available in the literature.

6 Conclusions

The nodes and weights computed with our two-stage algorithm are well suited for applications because of the regular distribution of the nodes. The theoretical results show that for the unit sphere equally weighted quadrature formulae cannot be found except for some special cases. Our weights are, however, very close to being equal.

Finding weights for all numbers of nodes (not only for those coinciding with the dimension of a polynomial space) will be one aim of further research. Another approach might be to inspect other integration measures and thus perhaps find equally weighted quadratures corresponding to these measures. Seymour and Zaslavsky [22] gave already a hint in this direction.

Our computed facility locations and weights are available at our WWW-server `http://www.mathematik.uni-dortmund.de/lsx/fliege/nodes.html`.

7 Appendix

In this section we give detailed numerical results concerning computation times, optimal function values and integration errors.

7.1 Computation Times

N	SA	L-BFGS	Stage 2
100	0.35	0.13	0.13
121	0.52	0.25	0.12
125	0.65	0.35	0.11
144	0.85	0.36	0.10
169	1.33	0.25	0.11
196	2.01	0.66	0.13
225	2.98	0.70	0.16
256	3.83	1.03	0.21
289	6.10	1.61	0.26
324	7.54	3.05	0.35
361	10.31	3.36	0.45
400	13.88	3.08	0.58
441	18.44	3.99	0.73
484	24.18	4.31	0.95
529	31.36	4.33	1.20
576	40.25	7.10	1.51
625	51.17	7.27	1.91
676	64.43	19.58	2.36
729	80.43	23.31	2.93
784	99.65	11.17	3.61
841	122.58	20.85	4.41
900	149.76	24.83	5.35
961	181.82	18.30	
1024	219.41	31.45	
1089	263.28	35.55	
1156	314.25	32.83	
1225	373.24	38.8	
1296	386.07	40.95	
1369	454.20	56.45	
1444	532.18	68.51	
1521	621.12	100.81	
1600	722.08	77.88	

Table 2: Computation times (in minutes) for Phase I (SA), Phase II (L-BFGS) and Stage 2.

7.2 Norm of Gradient

N	before L-BFGS	after L-BFGS
100	1.87952e+00	1.03521e-05
121	2.30062e+00	1.39569e-05
125	2.70879e+00	1.71996e-05
144	2.45123e+00	2.26959e-05
169	3.78806e+00	3.22294e-05
196	3.54435e+00	2.89859e-05
225	7.01217e+00	2.73103e-05
256	5.59760e+00	5.04821e-05
289	1.07360e+01	5.75318e-05
324	1.00568e+01	6.84213e-05
361	7.72356e+00	9.47381e-05
400	1.25450e+01	1.13484e-04
441	1.16049e+01	2.49419e-04
484	1.18645e+01	2.19365e-04
529	1.39922e+01	1.75517e-04
576	1.78617e+01	2.83658e-04
625	1.43587e+01	2.72287e-04
676	1.54297e+01	2.52806e-04
729	2.02775e+01	3.09822e-04
784	2.40794e+01	5.86850e-04
841	1.87958e+01	6.07594e-04
900	2.91767e+01	8.14847e-04
961	2.44264e+01	5.15149e-04
1024	2.46215e+01	7.58699e-04
1089	3.61267e+01	1.07609e-03
1156	3.20858e+01	6.84685e-04
1225	5.83467e+01	1.06225e-03
1296	4.23138e+01	1.06957e-03
1369	5.01770e+01	8.60001e-04
1444	6.15702e+01	1.16669e-03
1521	6.10294e+01	2.09635e-03
1600	8.62186e+01	1.73162e-03

Table 3: Norm of projected gradient before and after Phase II.

7.3 Optimal Function Values

N	opt. fct. val.	N	opt. fct. val.	N	opt. fct. val.
1	0.500000000000000	51	1145.421980634601	101	4633.736565899113
2	1.732050807568877	52	1191.931584709404	102	4727.836616833582
3	3.674234614174767	53	1239.371192268863	103	4822.876522749370
4	6.474691494688166	54	1287.777027461126	104	4919.017020539754
5	9.985281374238577	55	1337.095348267226	105	5015.984595705453
6	14.45297741422138	56	1387.383229252853	106	5113.980857754619
7	19.67528786123286	57	1438.618250640449	107	5212.872590887425
8	25.75998653126991	58	1490.774386078146	108	5312.735079920440
9	32.71694946014845	59	1543.835099598554	109	5413.598686896172
10	40.59645050819166	60	1597.941830199009	110	5515.409738229780
11	49.16525305762890	61	1652.942014269376	111	5618.196174615613
12	58.85323061173346	62	1708.879681503314	112	5721.843254029606
13	69.30636329662667	63	1765.802577927331	113	5826.594788317777
14	80.67024411429442	64	1823.667960263912	114	5932.188976096309
15	92.91165530254683	65	1882.441525304374	115	6038.834145813815
16	106.0504048286222	66	1942.122700405566	116	6146.342446580276
17	120.0844674474937	67	2002.874701748871	117	6254.947805249175
18	135.0894675567174	68	2064.536066225271	118	6364.363809130771
19	150.8815683337618	69	2127.100901550687	119	6474.845754099968
20	167.6416223992759	70	2190.693812716117	120	6586.250208760636
21	185.2875361493091	71	2255.001190975089	121	6698.613051331485
22	203.9301906628859	72	2320.633883745415	122	6811.932421612654
23	223.3470740518098	73	2387.072981838508	123	6926.224067750290
24	243.8127602988214	74	2454.369689041121	124	7041.576937871167
25	265.1333263173680	75	2522.674871841452	125	7157.725026469192
26	287.3026150330422	76	2591.850152353984	126	7274.905317855259
27	310.4915423582046	77	2662.047213292928	127	7393.080343399315
28	334.6344399204439	78	2733.248357479150	128	7512.107319268834
29	359.6039459039517	79	2805.437319233963	129	7632.308179538056
30	385.5308380633188	80	2878.528532667672	130	7753.231606916379
31	412.2612746505304	81	2952.574784715881	131	7875.202162294806
32	440.2040574480203	82	3027.592457041952	132	7998.180015680348
33	468.9048532837587	83	3103.492434468477	133	8122.089721194863
34	498.5698724906699	84	3180.361442939105	134	8247.026135444601
35	529.1224083754304	85	3258.213663077457	135	8372.990228476745
36	560.6188877311326	86	3337.002642986308	136	8499.534494782418
37	593.0385035664556	87	3416.720196759413	137	8627.406389900463
38	626.3890090168368	88	3497.439018624774	138	8756.227056953609
39	660.6752788346454	89	3579.128462215714	139	8886.189301095805
40	695.9167443420025	90	3661.730604239242	140	9016.677941858294
41	732.0781075436930	91	3745.291636241583	141	9148.320715713384
42	769.1908464592542	92	3829.844338421653	142	9280.915900377935
43	807.1742630846396	93	3915.424609921690	143	9414.540733412508
44	846.1884010611068	94	4001.771675565426	144	9548.995540368338
45	886.1714324245660	95	4089.329376437116	145	9684.381825577339
46	927.0722245627560	96	4177.533599623282	146	9821.080438306173
47	968.7134553438224	97	4266.822464156598	147	9958.667725318692
48	1011.557182653625	98	4357.268650942716	148	10096.94635930301
49	1055.182314726305	99	4448.410420647641	149	10236.43577081063
50	1099.819290319058	100	4540.658785474170	150	10376.65239705270

Table 4: Optimal function values for $N = 1, \ldots, 150$

N	opt. fct. val.	N	opt. fct. val.
151	10517.97837624036	225	23449.50788900559
152	10660.27011040682	256	30507.28651936315
153	10803.39671662756	289	39048.39864813881
154	10947.64768617304	324	49268.55128304187
155	11093.06775325911	361	61373.67131670026
156	11239.01579509413	400	75583.41683491136
157	11386.08198597265	441	92126.95738137892
158	11534.21667076679	484	111248.9368335901
159	11683.39826474392	529	133202.4341035277
160	11833.29114316029	576	158253.8549244064
161	11984.14573100214	625	186684.1818973242
162	12136.13800011864	676	218782.7025117617
163	12289.00614464313	729	254850.5826315200
164	12442.84161574589	784	295204.0315406995
165	12597.79894491279	841	340171.5969625181
166	12753.56052792666	900	390088.4611362188
167	12910.44258184769	961	445306.6367858270
168	13068.14219473437	1000	482533.9014834165
169	13226.72433227355	1024	506190.2541103970
170	13386.37582221102	1089	573111.6049805244
171	13547.29447123441	1156	646459.4991660330
172	13708.88286327617	1225	726630.5007997637
173	13871.39313458218	1296	814037.6356036495
174	14035.00215609037	1369	909099.3208944425
175	14199.52017335327	1444	1012257.092371517
176	14364.90419085466	1521	1123952.210768201
177	14531.48287836277	1600	1244646.673002645
178	14698.77400378278		
179	14867.25730078455		
180	15036.57787263741		
181	15207.00129955274		
182	15378.22850973948		
183	15550.68605915045		
184	15723.84688160236		
185	15897.89806865325		
186	16072.97518632155		
187	16249.53844362568		
188	16426.55869680151		
189	16604.76043954694		
190	16783.59287979203		
191	16963.65583303697		
192	17144.79813650427		
193	17326.87226084459		
194	17509.64611329630		
195	17693.64129079832		
196	17878.54576471418		
197	18064.30096947719		
198	18251.19478087154		
199	18439.01909926687		
200	18627.84462660964		

Table 5: Optimal function values for $N = 151, \ldots, 200$ and $N = (m+1)^2$, $m = 15, \ldots, 39$

7.4 Integration Errors

N	$E_N(1)$	$E_N(2)$	$E_N(3)$
4	8.78020512761935e-02	8.02314577142333e-02	1.63077771006487e+01
9	1.82568386145073e-03	3.66400055561377e-02	5.86374220206978e+00
16	6.47555435502245e-05	1.33357778654557e-02	3.84578276300764e-01
25	1.14637638461091e-05	1.04260273744346e-02	6.36865828283532e+00
36	1.86766413734893e-06	5.30620344001739e-03	1.30216896104055e+00
49	3.63375059391151e-07	3.22585543364276e-03	3.29377086149765e-01
64	1.55262123187976e-08	1.82281833087059e-03	7.75684906587689e-02
81	6.34243844352131e-09	3.99734397182933e-03	3.04808623474023e-02
100	1.79390049780681e-10	5.25849065329808e-04	1.41877601874449e-02
121	3.82323232559047e-10	5.88276341729074e-04	5.52949345488457e-01
144	3.09989000267251e-12	2.07575081462397e-04	1.47101448488389e-02
169	6.30732424359995e-12	2.09911040692906e-03	6.09639002831513e-02
196	7.50037217567543e-12	3.06019614923425e-04	9.05882554830993e-04
225	1.73948892645188e-12	4.91320980190064e-04	1.20687438486979e-04
256	6.84448647313730e-12	1.73305492553497e-04	1.56308276753464e-04
289	3.41279678320842e-12	2.67638547107841e-04	3.21968300798490e-06
324	1.66529234544378e-11	5.34420107374955e-04	6.10158714887304e-06
361	1.98658355203907e-11	7.66889840053591e-04	1.92614169441722e-04
400	6.89803500756985e-12	2.36697999322948e-04	6.81244732083705e-06
441	2.48930255932451e-11	9.82449785513239e-05	1.19501387163813e-05
484	1.38707473032419e-11	2.06755141886810e-04	2.91858782146154e-07
529	6.37104588165120e-12	4.42939094332322e-05	4.00712006350030e-08
576	9.32012800760240e-13	1.91501913045550e-04	1.03361396518929e-08
625	1.79179656582764e-11	1.42458093476012e-04	9.88065539623172e-09
676	4.73172289852200e-12	3.80986094058053e-05	7.53528653782998e-12
729	1.14729132426078e-11	3.44813409482371e-04	9.73896221093302e-11
784	1.51374663922924e-11	8.52912954201780e-05	1.48357525184346e-10
841	2.92236375490865e-12	6.52532086018932e-05	1.23810910488784e-10
900	7.28427756803844e-12	5.37107738084881e-05	9.09644413026138e-11

Table 6: Integration errors for the functions f_1, f_2 and f_3

N	$E_N(4)$	$E_N(5)$	$E_N(6)$
4	8.63863610691049e+00	2.43847551902481e-02	4.20466354511539e-08
9	3.58165437871609e+00	1.27323380999543e-03	5.35772737786377e-07
16	1.72990192038830e+00	7.36360635752114e-06	1.48561718482654e-13
25	1.10573990177018e+00	4.39868345145060e-06	1.38117859163622e-12
36	6.35707342697431e-01	6.84682850057724e-07	4.20277007640024e-12
49	3.73302164375069e-01	7.71774822678631e-09	2.84492053395993e-12
64	2.75383881890808e-01	5.90205678104757e-10	3.30309564422482e-13
81	2.02667808758315e-01	2.31948588798633e-11	1.14544138063444e-12
100	1.38014529657397e-01	1.46518028986390e-12	6.32666141697769e-12
121	6.25631009615863e-02	7.17590721063136e-12	5.98681260816081e-12
144	7.43150766696460e-02	3.46370015819070e-12	3.71756445072258e-12
169	5.10474303893537e-02	1.71092065044119e-12	1.03226151584868e-12
196	3.39172073047075e-02	1.84395483487510e-12	8.74979006349613e-12
225	2.49789864238342e-02	1.15352425743328e-12	2.97533048546073e-12
256	2.19911962937554e-02	3.83994873767467e-12	1.73062736016916e-12
289	1.56354421898306e-02	6.24984013472116e-12	8.19204079571811e-12
324	1.36341507236423e-02	6.08577265952418e-12	1.50718239971814e-12
361	4.66955744375706e-03	6.13701367450505e-12	5.05528529193972e-12
400	3.80442052179990e-03	1.32553048143011e-12	5.92020562347612e-12
441	5.79717804994800e-02	2.79612355456664e-12	5.30813605384721e-11
484	6.48412780706010e-04	1.40311370833777e-11	2.33229754953435e-12
529	3.92340373280540e-05	8.82740752914637e-12	5.25164991788812e-12
576	1.04048908297590e-03	1.35463922702957e-11	1.31402007330639e-12
625	1.26821610672737e-02	1.45763847850404e-11	3.48331086197362e-12
676	1.03221549963817e-03	6.54489696511572e-12	1.30929616107278e-11
729	1.44645113579436e-03	1.39043576707255e-11	1.51104189924367e-11
784	3.01566112313209e-02	1.21756349775210e-11	3.41100122322816e-11
841	1.32422700519603e-05	3.69392387338645e-12	4.87604336248115e-12
900	5.55866037228356e-04	4.02290081228948e-12	3.74510112591958e-12

Table 7: Integration errors for the functions f_4, f_5 and f_6

References

[1] E. Bannai, R. M. Damerell. Tight spherical designs I. *Journal Mathematical Society Japan*, 31:199–207, 1979.

[2] E. Bannai, R. M. Damerell. Tight spherical designs II. *Journal London Mathematical Society*, 21(2):13–30, 1980.

[3] L. Bos. Some remarks on the Fejér-problem for Lagrange interpolation in several variables. *Journal Approximation Theory*, 60:133–140, 1990.

[4] Richard H. Byrd, Peihuang Lu, Jorge Nocedal and Ciyou Zhu. A limited memory algorithm for bound constrained optimiza-

tion. *SIAM Journal on Scientific Computing*, 16(5), 1995. Also published as Technical Report NAM-08, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois, USA. May 1994. Electronically distributed as `ftp://eecs.nwu.edu/pub/lbfgs/lbfgs_bcm/byrd-lu-nocedal-zhu.ps`

[5] Richard H. Byrd, Jorge Nocedal and Robert B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. Technical Report NAM-03, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois, USA. January 21, 1996. Electronically distributed as `ftp://eecs.nwu.edu/pub/lbfgs/lbfgs_bcm/byrd-nocedal-schnabel.ps`

[6] R. Cools. The construction of cubature formulae using invariant theory and ideal theory. Ph. thesis, Leuven 1989.

[7] J. Cui and W. Freeden. Equidistribution on the sphere. Bericht der Arbeitsgruppe Geomathematik, Nr. 142, Universität Kaiserslautern, 1995.

[8] P. Delsarte, J. M. Goethals and J. J. Seidel. Sperical codes and designs. *Geometriae Dedicata*, 6:363–388, 1977.

[9] Murray Dow. CONMIN code. Electronically distributed via `ftp://anusf.anu.edu.au/mld900/constr_minimum`

[10] William L. Goffe, Gary D. Ferrier and John Rogers. Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, 60:65–99, 1994.

[11] William L. Goffe, Gary D. Ferrier and John Rogers. SIMANN code. Electronically distributed as `ftp://netlib2.cs.utk.edu/opt/simann.f`

[12] Lester Ingber. ASA Version 10.9. Code and accompanying documentation. Electronically distributed via `http://www.ingber.com/#ASA-CODE` and `ftp://ftp.ingber.com`

[13] V. I. Krylov. Approximate calculation of integrals. ACM Monograph Series, Macmillan New York, London, 1962.

[14] J. R. Morris, D. M. Deaven and K. M. Ho. Genetic-algorithm energy minimization for point charges on a sphere. *Physical Review B*, 53(4):1740–1743, 1996.

[15] Stephen G. Nash. Newton-type minimization via the Lanczos method. *SIAM Journal on Numerical Analysis*, 21(4):770–778, 1984.

[16] M. Reimer. Constructive theory of multivariate functions. BI Wissenschaftsverlag, Mannheim, Wien, Zürich 1990.

[17] M. Reimer. On the existence-problem of Gauss-quadrature on the sphere. In: B.Fuglede e.a., eds., Approximation by solutions of partial differential equations, Kluwer 1992, 169-184.

[18] M. Reimer. On the existence of Gauss-like node-distributions on high-dimensional spheres. In: K. Jetter and F. Utreras, eds., Multivariate Approximation, World Scientific 1993, 281–291.

[19] M. Reimer. Quadrature rules for the surface integral of the unit sphere on extremal fundamental systems. *Mathematische Nachrichten*, 169:235–241, 1994.

[20] M. Reimer, B. Sündermann. A Remez-type algorithm for the calculation of extremal fundamental systems for polynomial spaces on the sphere. *Computing* 37:43–58, 1986.

[21] J. J. Seidel. Harmonics and Combinatorics. In: R. A. Askey. Special functions: Group theoretical aspects and applications, 287-303, D.Reidel Publishing Company 1984.

[22] P. D. Seymour, T. Zaslavsky. Averaging sets: A generalization of mean values and spherical designs. *Advances Mathematics*, 52:231–240, 1984.

[23] J. Steinacker, E. Thamm and U. Maier. Efficient integration of intensity functions on the sphere. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 56(1):97–107, 1996.

[24] A. H. Stroud, D. Secrest: Gaussian quadrature formulas. Prentice Hall Inc., Series in Automatic Computation, 1966.

[25] Yanmu Zhou. Arrangements of points on the sphere. Ph. thesis, Tampa, Florida, 1995.

[26] Ciyou Zhu. L-BFGS FORTRAN code. November 1994. Electronically distributed via `ftp://eecs.nwu.edu/pub/lbfgs/lbfgs_bcm/` and `http://www.eecs.nwu.edu/~ciyou/code/lbcode.html`

[27] Ciyou Zhu, Richard H. Byrd, Peihuang Lu and Jorge Nocedal. L-BFGS-B: a limited memory FORTRAN code for solving bound constrained optimization problems. Technical Report, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois, USA. 1994. Electronically distributed as `ftp://eecs.nwu.edu/pub/lbfgs/lbfgs_bcm/zhu-byrd-lu-nocedal.ps`

[28] Ciyou Zhu, Richard H. Byrd, Peihuang Lu and Jorge Nocedal. L-BFGS-B — FORTRAN subroutines for large-scale bound constrained optimization. Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois, USA. December 31, 1994. Electronically distributed as `ftp://eecs.nwu.edu/pub/ciyou/pp9.ps` or via `ftp://eecs.nwu.edu/pub/lbfgs/lbfgs_bcm/` and `http://www.eecs.nwu.edu/~ciyou/index.html`