Closing the Case $t = 3$ for 3-D Spherical t-Designs Using a Result-Verifying Nonlinear Solver*

Thomas Beelitz
Viadico AG, Sendlinger Straße 18, D-80331 München, Germany†

Bruno Lang
University of Wuppertal, Institute for Applied Computer Science and Scientific Computing, D-42097 Wuppertal, Germany
lang@math.uni-wuppertal.de

Peer Ueberholz
Niederrhein University of Applied Science, Department of Electrical Engineering and Computer Science, Reinarzstr. 49, D-47805 Krefeld, Germany
peer@ueberholz.de

Paul Willems
University of Wuppertal, Institute for Applied Computer Science and Scientific Computing, D-42097 Wuppertal, Germany
willems@math.uni-wuppertal.de

Abstract

The question if there exists an N-point spherical t-design is not yet settled for all combinations of t and N. Using our framework SONIC for the solution of nonlinear systems, we were able to close the two remaining open cases for $t = 3$. More precisely, a computational proof revealed that there are no spherical 3-designs with $N = 7$ or $N = 9$ points. We describe how these results were obtained and comment on the open cases for larger values of t.

Keywords: interval analysis, quadrature on the sphere, nonlinear algebraic systems
AMS subject classifications: 65D32, 65H10, 65G20

*Submitted: December 5, 2008; Revised: January 9, 2009; Accepted: July 1, 2009.

†T.B. contributed to this work while he was with the Institute for Applied Computer Science and Scientific Computing at the University of Wuppertal.

$N=$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
$t=1$	n	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	...
$t=2$	n	n	n	y	n	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	...
$t=3$	n	n	n	n	n	y	**?**	y	**?**	y	y	y	y	y	y	y	y	y	y	y	y	y	y	...
$t=4$	n	n	n	n	n	n	n	n	n	**?**	**?**	y	**?**	y	**?**	y	y	y	y	y	y	y	y	...
$t=5$	n	n	n	n	n	n	n	n	n	n	n	y	**?**	**?**	**?**	y	**?**	y	**?**	y	**?**	y	y	...

Figure 1: Known results for small t [13]. "y" means that a spherical t-design is known to exist for the respective combination (t, N), "n" indicates that it provably does not exist, and "**?**" marks an open case.

1 Introduction

Let S_d denote the unit sphere in d dimensions, i.e.,

$$S_d = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 = 1\}.$$

An *N-point spherical t-design* is an arrangement of N points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ on the sphere S_d such that the unit-weight quadrature rule

$$\int_{S_d} p(\mathbf{x})\, d\mu \quad \approx \quad \frac{\sigma_d}{N} \sum_{i=1}^{N} p(\mathbf{x}_i) \tag{1}$$

is *exact* for all polynomials $p = p(x_1, \ldots, x_d)$ with total degree $\leq t$. Here, μ denotes the unit measure, and σ_d is the area of the sphere. In the following we will focus on the case $d = 3$, where $\sigma_d = 4\pi$.

Spherical designs have close connections to high-order numerical cubature for the sphere (with applications in, e.g., molecular sciences [20]), to statistics [7], and to algebra and coding theory [18, 21].

It is known that 3D spherical t-designs cannot exist if the number of points is too small. More precisely, we must have [2, 3, 10]

$$N \quad \geq \quad \begin{cases} \dfrac{(t+1)(t+3)}{4}, & \text{if } t \text{ is odd,} \\ \dfrac{(t+2)^2}{4}, & \text{if } t \text{ is even,} \end{cases}$$

and these lower bounds can be increased by one if $t \neq 1, 2, 3, 5$. On the other hand, constructive methods prove the existence of spherical t-designs for sufficiently large N (see, e.g., [17]). In between these two extremes, the existence or non-existence of spherical t-designs has been proved for various combinations (t, N), relying on different techniques including interval computations; cf. [13]. Nevertheless, open cases remain even for $t = 3$. The situation for $t \leq 5$ is summarized in Fig. 1.

With the present paper we close the case $t = 3$. More precisely, we use result-verifying methods to prove that spherical 3-designs with $N = 7$ or $N = 9$ points do not exist.

The remainder of the paper is organized as follows. In Section 2 we will show how N-point spherical t-designs are related to suitable nonlinear systems. Section 3 contains the result for the $(3, 7)$ case and a description of our nonlinear solver SONIC

with which this case was solved. The nonlinear system corresponding to the $(3, 9)$ case could be handled only with an efficiently parallelized version of the solver, as explained in Section 4. A discussion on the perspectives for larger t will close the paper.

2 Spherical t-designs and nonlinear systems

In three dimensions, (1) being exact for all polynomials p of total degree $\leq t$ is equivalent to

$$\int_S x^k y^\ell z^m \, d\mu \quad = \quad \frac{4\pi}{N} \sum_{i=1}^{N} x_i^k y_i^\ell z_i^m \quad \text{for all } k,\, \ell,\, m \text{ s.t. } k+\ell+m \leq t. \tag{2}$$

The left-hand sides of (2) can be evaluated exactly. Using spherical coordinates $(x, y, z) = (\sin\varphi\cos\theta, \sin\varphi\sin\theta, \cos\varphi)$ with $\varphi \in [0, \pi]$ and $\theta \in [0, 2\pi]$, a straightforward but somewhat longish manipulation using the formulae in [8] yields

$$\begin{aligned} \int_S x^k y^\ell z^m \, d\mu &= \int_0^{2\pi} \int_0^{\pi} (\sin\varphi\cos\theta)^k (\sin\varphi\sin\theta)^\ell (\cos\varphi)^m \cdot \sin\varphi \, d\varphi \, d\theta \\ &= 4\pi \cdot \gamma_{k,\ell,m}, \end{aligned}$$

where

$$\gamma_{k,\ell,m} \quad = \quad \begin{cases} \dfrac{o(k-1)\cdot o(\ell-1)\cdot o(m-1)}{o(k+\ell+m+1)}, & \text{if } k,\, \ell,\, m \text{ are all even} \\ 0, & \text{otherwise} \end{cases},$$

and

$$o(j) \quad = \quad 1 \cdot 3 \cdot \ldots \cdot j$$

denotes the product of the odd numbers up to j with $o(-1) := 1$.

Multiplying (2) by $N/4\pi$ and switching to spherical coordinates, $(x_i, y_i, z_i) = (\sin\varphi_i\cos\theta_i, \sin\varphi_i\sin\theta_i, \cos\varphi_i)$, N-point spherical t-designs thus are exactly the solutions of the nonlinear system

$$N\gamma_{k,\ell,m} \quad = \quad \sum_{i=1}^{N} (\sin\varphi_i\cos\theta_i)^k (\sin\varphi_i\sin\theta_i)^\ell (\cos\varphi_i)^m, \quad k+\ell+m \leq t, \tag{3}$$

lying in the box $\theta_i \in [0, 2\pi]$, $\varphi_i \in [0, \pi]$. (For $t = 3$, the only non-zero left-hand sides are $N\gamma_{0,0,0} = N$ and $N\gamma_{2,0,0} = N\gamma_{0,2,0} = N\gamma_{0,0,2} = N/3$.) This system is modified slightly to facilitate its solution with result-verifying methods.

1. The $(k = 0, \ell = 0, m = 0)$ equation reads "$N = N$" and is dropped. The remaining system contains $\frac{t}{6}(t^2 + 6t + 11)$ equations.
2. Given a spherical t-design, any rotation of the sphere yields another valid design. To remove this rotational symmetry, the first point is located at the north pole $(\theta_1 = 0, \varphi_1 = 0)$, and the second point is fixed to longitude $\theta_2 = 0$. Substituting these values directly into the equations reduces the number of variables to $2N-3$.
3. Given a spherical t-design, any renumbering of the points yields another valid design. To remove this symmetry, we enforce an (almost) fixed ordering by adding the $(N-2)(N-3)/2$ inequalities $\theta_i \leq \theta_j$ for $3 \leq i < j$. With this ordering, the point $\mathbf{x}_3$ must lie in the eastern hemisphere. Thus the range of θ_3 can be restricted to $[0, \pi]$.

Mathematically, the $N-3$ inequality constraints $\theta_i \le \theta_{i+1}$, $i = 3, \ldots, N-1$, are sufficient, due to the transitivity of "$\le$". However, we do not remove the remaining inequalities because they may allow us to remove boxes at a higher recursion level. To give a simple example, a box with $(\theta_3, \theta_4, \theta_5) \in [\pi, 3\pi/2] \times [\pi/2, \pi] \times [0, \pi/2]$ may be not be discarded if we only have the constraints $\theta_3 \le \theta_4$ and $\theta_4 \le \theta_5$ (since the ranges of the variables are not disjoint), whereas the additional constraint $\theta_3 \le \theta_5$ would allow discarding the box. For similar reasons we do not apply the substitution $z^2 = 1 - x^2 - y^2$, which would reduce the number of equations to $t(t+2)$.

Finally, subtracting the right-hand sides from the (in)equalities leads to a system of the form

$$\begin{aligned} \mathbf{f}(\mathbf{z}) &= \mathbf{0}, \\ \mathbf{g}(\mathbf{z}) &\le \mathbf{0}, \end{aligned}$$

where $\mathbf{z} \in [0, \pi] \times [0, 2\pi]^{N-3} \times [0, \pi]^{N-1}$ contains the non-fixed angles θ_i and φ_i. For arbitrary t and N, the system can be set up in a completely automated way.

In the following section we briefly describe our software SONIC, which is targeted at the verified solution of such inequality-constrained systems.

3 Solving moderately difficult systems

As pointed out in the preceding section, to tackle spherical t-designs with N points we must find all solutions for a nonlinear system over $n = 1 + (N-3) + (N-1)$ variables z_i with given initial ranges $[\mathbf{z}]^{(0)} = [0, \pi] \times [0, 2\pi]^{N-3} \times [0, \pi]^{N-1}$, $m = \frac{t}{6}(t^2 + 6t + 11)$ equality constraints $\mathbf{f}(\mathbf{z}) = \mathbf{0}$, and $k = (N-2)(N-3)/2$ inequality constraints $\mathbf{g}(\mathbf{z}) \le \mathbf{0}$.

The crucial point for our application is that we need *reliable* results, that is, the methods used may never lose a solution to the system, even in the face of (inevitable) rounding errors.

We use interval arithmetic [15, 16]. Most modern hardware platforms support *directed rounding* modes for floating-point operations. By using the rounding modes appropriately one can compute bounds $[f]([\mathbf{z}])$ on the range of a (continuous) function $f : \mathbb{R}^n \to \mathbb{R}$ over a (compact) real interval $[\mathbf{z}] \in \mathbb{IR}^n$. These bounds are indeed *verifiably correct* in the sense that they are guaranteed to contain the exact range $f([\mathbf{z}])$. In principle, such bounds alone are sufficient for the verified solution or optimization of arbitrary nonlinear systems. A very general technique suitable (among other things) for the solution of general nonlinear systems or global optimization is the Branch-and-Prune approach, which is summarized in Fig. 2.

The remainder of this section is structured in two parts. First we will give a short overview of our custom verified solver. Based on this overview we will then present and discuss our first set of results for spherical t-designs, which close the case $t = 3, N = 7$.

3.1 The SONIC software framework

In an ongoing research project, which grew out of an application for the design of robust dynamical systems [19], we are developing the custom software tool SONIC [4, 5, 22] (*Solver and Optimizer for Nonlinear Systems based on Interval Computations*). At its core we run a basic Branch-and-Prune engine similar to the one shown in Fig. 2. In the following we give a coarse overview of the contraction techniques implemented in our solver; more details are given in [6].

$\mathcal{A} := \{ [\mathbf{z}]^{(0)} \}$ { list of "active" subboxes of $[\mathbf{z}]^{(0)}$ }
$\mathcal{L} := \emptyset$ { list of subboxes that might contain solutions }
while $\mathcal{A} \neq \emptyset$
    remove one box $[\mathbf{z}]$ from $\mathcal{A}$
    { PRUNE }
    use contraction techniques to discard parts of $[\mathbf{z}]$ which
    provably cannot contain a solution of the system
    if $\max\{\mathrm{diam}([z_i])\} \leq tol$ **then**
        insert $[\mathbf{z}]$ into $\mathcal{L}$
    else
        { BRANCH }
        subdivide the box $[\mathbf{z}]$ along one or more of its coordinates into
        smaller boxes and insert these into $\mathcal{A}$
    end if
end while

Figure 2: The basic Branch-and-Prune algorithm.

Symbolic Techniques. Internally, functions are represented symbolically, that is, as *term trees*. Each node represents an atomic operation (like +, /, pow, ln, sin, ...) or a variable. The trees of all function terms of the nonlinear system are fused together into a single directed acyclic graph, the *constraint network*, by merging nodes that represent common subterms in the system. For each variable there is exactly one node in the network without ingoing edges (a source). This handling of common subterms is especially fruitful in the context of spherical designs, as e.g., subterms such as $(\sin\varphi_i \cos\theta_i)^2$ occur multiple times in the system (3).

The simplest symbolic technique is the evaluation of a function term on a box, also called *natural interval evaluation*. Given the constraint network, natural interval evaluation corresponds to a forward sweep from the variables (leaves) to the root, evaluating each encountered elementary operation with interval arithmetic.

A more general symbolic contraction scheme is *constraint propagation* (CP), see e.g., [14]. Each internal node in the constraint network corresponds to an intermediate result in the evaluation of at least one function in the nonlinear system. The node is assigned an internal interval variable, which holds the best known bounds on the intermediate result and is initialized to $\mathbb{R}$. Then we iteratively improve these bounds (for all or some nodes of the net, in a suitable ordering) by using the elementary operators to contract the (internal or not) variables connected to a node until the changes become smaller than a given threshold.

The advantages of CP over the numerical techniques described below are twofold: It is faster since its complexity is linear in the number of elementary operators in the system, and it can be formulated to work on arbitrary real sets. The latter property makes it applicable to systems with non-differentiable, non-continuous, or even functions that are not defined everywhere.

Numeric Techniques. Consider a single equality constraint $f(\mathbf{z}) = 0$ and a box

$[\mathbf{z}] \in \mathrm{I}\mathbb{R}^n$ on which the (scalar) function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable. From the term tree for f we can, using symbolic derivatives, construct term trees for the gradient ∇f and for the Hessian H_f of f. Based on these we can formulate order-1 or order-2 Taylor expansions of f around a point $\mathbf{c} \in [\mathbf{z}]$ to get additional constraints

$$0 = f(\mathbf{c}) + \nabla f([\mathbf{z}])([\mathbf{z}] - \mathbf{c}), \tag{4}$$

for an order-1 expansion and

$$0 = f(\mathbf{c}) + \left(\nabla f(\mathbf{c}) + ([\mathbf{z}] - \mathbf{c})^T H_f\right)([\mathbf{z}] - \mathbf{c}), \tag{5}$$

for order 2. Any solution $\mathbf{z}$ to $f(\mathbf{z}) = 0$ must also satisfy (4) and (5). Therefore, the symbolic techniques described earlier can be applied to (4) and/or (5) to obtain a sharper enclosure $[f]([\mathbf{z}])$ or to contract the box $[\mathbf{z}]$. We call this *Taylor-1* and *Taylor-2* contraction, respectively.

Taking a step back and looking again at the complete system $\mathbf{f}(\mathbf{z}) = \mathbf{0}$, where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, we analogously can deduce that any solution of the system must also be a solution of the interval linear system

$$(R \cdot J_{\mathbf{f}}([\mathbf{z}])) \cdot ([\mathbf{z}] - \mathbf{c}) = -R \cdot \mathbf{f}(\mathbf{c}), \tag{6}$$

where $J_{\mathbf{f}}$ is the Jacobian of $\mathbf{f}$, and $R \in \mathbb{R}^{m \times m}$ is a suitable preconditioner. Now any algorithm for the solution of interval linear systems can be employed to contract $[\mathbf{z}]$ based on (6), for example one or more steps of the Gauss–Seidel iteration (see e.g. [16]). We call the resulting contractor *(Interval) Newton*. The performance and cost of this method strongly depend on the choice of the preconditioner R. A popular choice is the inverse midpoint preconditioner

$$R \;=\; \Big[\operatorname{mid} J_{\mathbf{f}}([\mathbf{z}])\Big]^{-1},$$

which gives good contractions at a reasonable runtime overhead. (For non-square or very ill-conditioned systems a suitable subset of the equations and/or variables must be selected prior to inverting.) Even better results can be achieved with optimal linear programming preconditioners [4, 16], but the incurred cost in runtime makes them inefficient in some situations.

In its default configuration, SONIC uses constraint propagation heavily, coupled with Taylor-1 and Interval Newton. For the latter, a linear programming preconditioner is used, but only as backup, if the contraction resulting from the inverse midpoint preconditioner was not satisfactory.

The development of SONIC is continued, and the software will be made publicly available after a major code restructuring. Interested readers may obtain the current version by sending an email to `SONIC@math.uni-wuppertal.de`.

There are many other excellent software tools, which could possibly have been applied to solve the nonlinear systems resulting from the quest for spherical t-designs, for example, the GlobSol package [12] or the ALIAS library [1]. We decided to use SONIC because it had proved competitive to the other tools in a variety of test cases [4] and because preliminary tests indicated that a highly scalable parallel solver would be needed to tackle the $t = 3, N = 9$ problem; cf. Sect. 4.

3.2 Results for $t = 3, N \leq 8$.

We started out with running our solver on the nonlinear systems of the form (3) for the still open cases according to Fig. 1, beginning with $t = 3$.

Before we continue we want to make a few comments on the meaning of the phrase "verified" computing. Although we can prove, in a mathematical sense, that an algorithm is correct even in the face of rounding errors, we cannot exclude completely the possibility of bugs and faulty implementations (that is, human error) or hardware failures. In this sense, there can be no such thing as an absolutely verifed solver. Nevertheless, the probability of a software or hardware error to persist and influence the results *can* be driven arbitrarily close to zero by repeated testing with a good testset on different hardware platforms.

Our solver framework SONIC has been, and is, tested exhaustively with a large variety of test problems, e.g., from the COPRIN project [9]. The ramifications of the previous paragraph notwithstanding, we are confident that the (comparatively small) parts of the code, which are actually effecting the contraction of boxes, are indeed correct.

To bolster this claim and to validate our results in this work regarding spherical t-designs, we applied our solver to cases (t, N) where the exact solutions are known (see Fig. 1 again). As pointed out in Sect. 2, the nonlinear systems for spherical designs are all quite similar. Therefore any possible errors in the code which could endanger correctness would be likely to affect more than one case.

For all of the following test cases, recall the parameter *tol* from Fig. 2: The computation is stopped when all boxes have been discarded (then the system has no solution) or when none of the remaining boxes has an edge that is longer than *tol* (then these small boxes cover the solution set).

$t = 3, N = 4$**:** The start box is discarded immediately, which verifies the known result that there are no spherical 3-designs with 4 points.

$t = 3, N = 5$**:** After considering a total of 89 boxes, all boxes could be discarded. This verifies the known result that there are no spherical 3-designs with 5 points.

$t = 3, N = 6$**:** According to [13], the solutions form regular octahedrons in this case. In this case the steps we take to remove symmetry from the system (see Sect. 2) do not suffice to make the solution set discrete or even unique. Nevertheless, when run with $tol = 0.01$ and several different settings for the solver's internal switches, the remaining boxes always contained the complete solution set.

$t = 3, N = 8$**:** Hardin and Sloane give a particular solution for this case (see `http://www.research.att.com/$\sim$njas/sphdesigns`), which was always recovered by our solver.

Based on the confidence resulting from these testing safeguards, we ran our solver to find spherical 3-designs with 7 points. The default configuration terminated with an empty solution list after considering a total of 880 509 boxes, which is a computational proof of our first major result:

There are no spherical 3*-designs with* 7 *points.*

More detailed information about multiple runs with differing configurations and activated contraction techniques is given in Tab. 1. For the nonlinear systems resulting from spherical t-designs, the more expensive contraction techniques did not yield a significant reduction of the overall number of boxes to be considered and thus led to

Configuration	# Boxes	Time
Only symbolic CP ("fast")	1 051 611	2h 09m 47s
+ Taylor-1, Newton ("default")	880 509	15h 17m 14s
+ Taylor-2 ("pedantic")	1 731 083	95h 08m 18s

Table 1: Statistics of different runs for the $t = 3, N = 7$ problem. All computations were done on a 1.7GHz Pentium-4 system.

a heavy increase in the processing time, compared to the "fast" (CP only) version. Adding the Taylor-2 contractor can even increase the number of boxes, due to a different subdivision strategy. Based on these results, the experiments in the following sections were made with the "fast" version.

4 The case $t = 3$, $N = 9$: A difficult system

The application of our result verifying solver SONIC to the last open case of the spherical 3-designs, $N = 9$, turned out to be computationally very demanding. An efficient shared-memory parallelization with OpenMP and a Task Scheduling algorithm is described in [5]. However, the $t = 3$, $N = 9$ case requires even more computational power, and therefore an efficient distributed memory parallelization with MPI is needed. A static load balancing algorithm cannot be applied, since the time for analyzing a box completely varies widely, depending on the size of the box, the effectiveness of the contraction methods, and the size of the associated subtree.

In a first attempt we implemented a simple master–worker model [4], which is a centralized dynamic load balancing algorithm. In this ansatz a master process stores all active boxes. The master sends the boxes to the worker processes, which analyze them. If a box provably does not contain a solution then it is discarded. If a box may contain a solution and if it is small enough then the worker stores this box in a solution list. Otherwise the box is split into several subboxes, and these are sent back to the master process.

This algorithms scales quite well, but in our problem the number of active boxes increased very rapidly because "not enough" boxes could be discarded at the higher levels of the Branch-and-Prune tree (for more details see Tab. 2 and the discussion, both in Sect. 5). Therefore we ran out of memory on the master node.

This memory problem can be avoided using a nearest neighbor dynamic load balancing algorithm. In the context of a parallel Interval Newton algorithm, such methods were first applied by Gau and Stadtherr [11]. We modified the method such that the active boxes are distributed over all processes, arranged in a d-dimensional process grid. Each process manages its own worklist of active boxes. Neighboring processes frequently exchange the number of active boxes in their worklists, this number serving as a rough estimate for the computational load. If one process has much more load than one of its neighboring processes then they exchange load, i.e., boxes are sent from processes with a large number of boxes to processes with only a few boxes in their worklist. Almost all communication is done without explicit synchronization. In this way a good load balancing in computing time and memory usage can be achieved and idle processors are avoided, resulting in very good scaling behavior up to a large number of processors. Details of the algorithms are given in [22].

	$t=3, N=6$		$t=3, N=7$		$t=3, N=8$		$t=3, N=9$	
ε	$h_{\max}$	b_{cons}	$h_{\max}$	b_{cons}	$h_{\max}$	b_{cons}	$h_{\max}$	b_{cons}
3.2	3	5	4	7	5	9	6	11
1.6	12	667	15	2 489	18	8 201	21	24 905
0.8	21	11 225	26	266 571	31	3 702 869	36	35 795 105
0.4	30	14 381	37	954 337	44	77 466 455	51	> 1 G
0.2	39	14 605	48	1 051 579	57	316 947 331		
0.1	48	14 937	59	1 051 611	70	886 623 249		
h_{avg}	13.9		20.0		(≈ 30)		38.5	

Table 2: Number of boxes, b_{cons}, that had to be considered to reach a prescribed precision ε, and "average recursion height" h_{avg} for four problems with $t = 3$.

Based on the experience with the $t = 3$, $N = 7$ case, the $t = 3$, $N = 9$ problem was solved using only symbolic CP. This setting minimizes the time per box, at the cost of a higher number of boxes to be considered; cf. Tab. 1. The computations were done partly on the IBM Blue Gene at the University of Edinburgh and partly on the HP XEON-Cluster at the University of Wuppertal. The algorithm finished with an empty solution list after considering a total of 385 821 409 493 boxes. This implies that

There are no spherical 3*-designs with* 9 *points,*

closing the last open case for the $t = 3$ spherical t-design problem.

5 Discussion and perspectives

Given the results from the preceding sections, a natural question is whether the remaining open cases for $t = 4, 5, \ldots$ can be closed in a similar way. In our opinion the answer is "not yet", for the reasons explained below.

Table 2 gives, for four of the $t = 3$ problems, the overall number of boxes that had to be considered during the Branch-and-Prune algorithm in order to reach a prescribed precision ε, i.e., $\max_{i=1}^{n} \operatorname{diam}[z_i] \le \varepsilon$ for each resulting box.

For $\varepsilon = 3.2$ only the $\theta_4, \ldots \theta_N$ components of the starting box $[\mathbf{z}]^{(0)}$ must be subdivided (or tightened) because the other components are already small enough. Thus the precision $\varepsilon = 3.2$ is reached after at most $h_{\max} = N - 3$ bisection steps, and at most $b_{\max} = \sum_{k=0}^{h_{\max}} 2^k = 2^{h_{\max}+1} - 1$ boxes can be considered in a bisection tree of height $h_{\max}$. Halving ε can require one additional bisection in each direction, i.e., increase the maximum height $h_{\max}$ of the bisection tree by $2N - 3$.

The b_{cons} data show that at the highest levels of the tree, the number of boxes grows roughly as 1.65^{ℓ}, ℓ denoting the level; cf. the left picture in Fig. 3. When the boxes are small enough ($\varepsilon \lesssim 0.4$) then SONIC's elimination and tightening techniques become more effective, and the overall number of boxes is saturating, corresponding to an "average height" $h_{\text{avg}} = \log_2(\max b_{\text{cons}})$. These values are also given in Tab. 2.

If the trend carries over to $t = 4$ (with h_{avg} lower by approximately 4, as suggested by the data in Tab. 3 and the right picture in Fig. 3), then we would expect a value $h_{\text{avg}} \approx 46$ for the smallest open case, $N = 10$. The huge number of boxes, $2^{46} \approx 10^{14}$, together with the fact that for $t = 4$ considering a single box takes roughly three times longer than in the cases $t = 3$, implies that the $t \ge 4$ problems by far exceed the

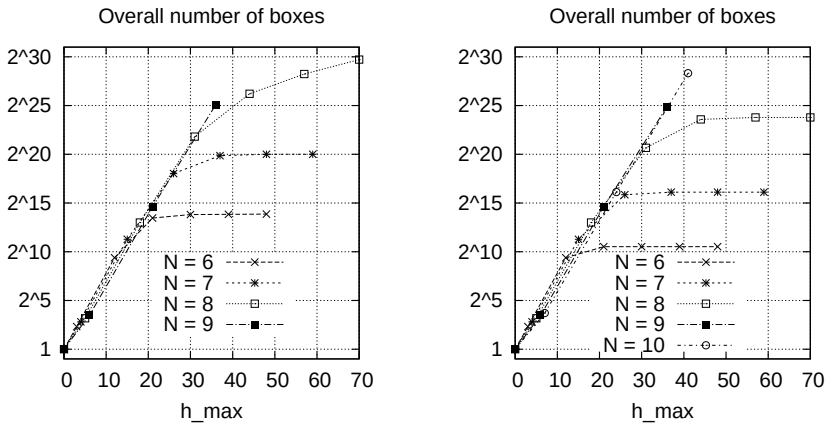

Figure 3: Overall number of boxes that had to be considered in the uppermost levels of the Branch-and-Prune tree for several problems ($t = 3$, left, and $t = 4$, right).

available computational ressources and that substantial theoretical and/or algorithmic progress is necessary before being able to solve them.

There might be hope, though. It seems that *a spherical t-design with an even number of points always contains at least one pair of diametrically opposite points on the sphere*. Since we already fix the first point to the north pole, we can implement this assumption by requiring that the second point not only has longitude $\theta_2 = 0$ but is fixed to the south pole ($\theta_2 = 0$, $\varphi_2 = \pi$).

In the $t = 3$, $N = 6$ case, this makes the solution indeed unique. For the resulting system, the solver finishes after working on just one box: the default contraction techniques are sufficient to contract the start box to $tol = 10^{-8}$ without any need for bisection. In the $t = 3$, $N = 8$ case, two different solutions remain. To cover these, a total of 5 792 291 boxes had to be considered for $\varepsilon = 10^{-8}$ — much less than the roughly 900 million without this restriction, cf. Tab. 2. In other cases the gain is not

	$t=4, N=6$		$t=4, N=7$		$t=4, N=8$		$t=4, N=9$		$t=4, N=10$	
ε	$h_{\max}$	b_{cons}	$h_{\max}$	b_{cons}	$h_{\max}$	b_{cons}	$h_{\max}$	b_{cons}	$h_{\max}$	b_{cons}
3.2	3	5	4	7	5	9	6	11	7	13
1.6	12	667	15	2 489	18	8 201	21	24 905	24	71 415
0.8	21	1 465	26	59 125	31	1 659 563	36	31 345 513	41	334 129 383
0.4	30	1 465	37	71 051	44	12 587 781	51	> 1 G	58	> 1 G
0.2	39	1 465	48	71 103	57	14 471 185				
0.1	48	1 465	59	71 103	70	14 471 201				
h_{avg}	10.5		16.1		23.8		(≈ 34)		(≈ 46)	

Table 3: Number of boxes, b_{cons}, that had to be considered to reach a prescribed precision ε, and "average recursion height" h_{avg} for five problems with $t = 4$.

that spectacular, but we expect substantial savings for the harder problems.

Thus a promising first step might be to prove the conjecture formulated above.

Acknowledgements

This work was partly carried out under the HPC-EUROPA project (RII3-CT-2003-506079), with the support of the European Community - Research Infrastructure Action under the FB6 "Structuring the European Research Area" program. Furthermore we would like to thank Peter Mättig and Torsten Harenberg from the University of Wuppertal for giving us access to their XEON cluster and the two unknown referees for their valuable comments.

References

[1] *ALIAS homepage*. `http://www-sop.inria.fr/coprin/logiciels/ALIAS`.

[2] E. Bannai and R. M. Damerell, *Tight spherical designs I*, J. Math. Soc. Japan, 31 (1979), pp. 199–207.

[3] ———, *Tight spherical designs II*, J. London Math. Soc., 21 (1980), pp. 13–30.

[4] T. Beelitz, *Effiziente Methoden zum verifizierten Lösen von Optimierungsaufgaben und nichtlinearen Gleichungssystemen*, PhD thesis, Bergische Universität Wuppertal, 2006. In German.

[5] T. Beelitz, C. Bischof, and B. Lang, *Efficient task scheduling in the parallel result-verifying solution of nonlinear systems*, Reliab. Comput., 12 (2006), pp. 141–151.

[6] T. Beelitz, A. Frommer, B. Lang, and P. Willems, *Symbolic-numeric techniques for solving nonlinear systems*, Proc. Appl. Math. Mech., 5 (2005), pp. 705–708.

[7] F. Bertrand, *Plans sphériques de force t et applications en statistique*, PhD thesis, Université Louis Pasteur (Strasbourg I), 2007.

[8] I. N. Bronstein and K. A. Semendjaev, *Taschenbuch der Mathematik*, Teubner Verlagsgesellschaft, Leipzig, Germany, 1979.

[9] *COPRIN homepage*. `http://www-sop.inria.fr/coprin`.

[10] P. Delsarte, J.-M. Goethals, and J. J. Seidel, *Spherical codes and designs*, Geom. Dedicata, 6 (1977), pp. 363–388.

[11] C.-Y. Gau and M. Stadtherr, *Parallel interval-Newton using message passing: Dynamic load balancing strategies*, in Proc. ACM/IEEE Conference on Supercomputing, ACM, New York, 2001.

[12] *GlobSol homepage*. `http://interval.louisiana.edu/GlobSol`.

[13] R. H. Hardin and N. J. A. Sloane, *McLaren's improved snub cube and other new spherical designs in three dimensions*, Tech. Rep., AT&T Bell Laboratories, 2002.

[14] E. Hyvönen, *Constraint reasoning based on interval arithmetic: The tolerance propagation approach*, Artificial Intelligence, 58 (1992), pp. 71–112.

[15] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis*, Springer-Verlag, London, 2001.

[16] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Dordrecht, 1996.

[17] J. Korevaar and J. L. H. Meyers, *Spherical Faraday cage for the case of equal point charges and Chebychev-type quadrature*, J. Integral Transforms and Special Functions, 1 (1993), pp. 105–127.

[18] W. Lempken, B. Schröder, and P. H. Tiep, *Symmetric squares, spherical designs, and lattice minima*, J. Algebra, 240 (2001), pp. 185–208.

[19] M. Mönnigmann, W. Marquardt, C. Bischof, T. Beelitz, B. Lang, and P. Willems, *A hybrid approach for efficient robust design of dynamic systems.*, SIAM Rev., 49 (2007), pp. 236–254.

[20] R. J. Morris, *An evaluation of spherical designs for molecular-like surfaces*, J. Mol. Graphics Modell., 24 (2006), pp. 356–361.

[21] G. Nebe, *Codes and invariant theory*, 2006. RWTH Aachen University. Unpublished survey for the Farewell Symposium for R. van der Waall, Universiteit van Amsterdam.

[22] P. Ueberholz, P. Willems, M. Bull, and B. Lang, *Non-blocking load balancing for branch-and-bound algorithms*, 2008. To appear in Proc. PARA 08.