Numerische Mathematik manuscript No.
(will be inserted by the editor)

Computational Existence Proofs for Spherical t-Designs

Xiaojun Chen[1], **Andreas Frommer, Bruno Lang**[2]

[1] Department of Applied Mathematics, Hong Kong Polytechnic University, Hong Kong maxjchen@inet.polyu.edu.hk
[2] Department of Mathematics, University of Wuppertal, 42097 Wuppertal, Germany {frommer,lang}@math.uni-wuppertal.de

Received: date / Revised version: date

Summary Spherical t-designs provide quadrature rules for the sphere which are exact for polynomials up to degree t. In this paper, we propose a computational algorithm based on interval arithmetic which, for given t, upon successful completion will have proved the existence of a t-design with $(t+1)^2$ nodes and will have computed narrow interval enclosures which are known to contain these nodes with mathematical certainty. Since there is no theoretical result which proves the existence of a t-design with $(t+1)^2$ nodes for arbitrary t, our method contributes to the theory because it was tested successfully for $t = 1, 2, \ldots, 100$, i.e., for all t considered so far. The t-design is usually not unique; our method aims at finding a well-conditioned one. The method relies on computing an interval enclosure for the zero of a highly nonlinear system of dimension $(t+1)^2$. We therefore develop several special approaches which allow us to use interval arithmetic efficiently in this particular situation. The computations were all done using the MATLAB toolbox INTLAB.

Key words spherical design, interval arithmetic, system of nonlinear equations

Mathematics Subject Classification (1991): 65H10, 65G20

1 Introduction

Finding "good" finite sets of points on the unit sphere S^d in the Euclidean space $\mathbb{R}^{d+1}$ has been a hot research topic in mathematics,

physics, and engineering for more than hundred years. In the late 90's, S. Smale ranked "the distribution of points on the 2-sphere" as one of 18 challenge problems for the 21st century [23]. There are several concepts of "good" finite sets of points on S^d for different purposes. A *spherical t-design* is considered a "good" finite set of points on S^d for global polynomial approximations on the sphere.

Definition 1 *A finite set $Y = \{y_1, \ldots, y_N\} \subset S^d$ is called a spherical t-design on S^d if for any polynomial $p : \mathbb{R}^{d+1} \to \mathbb{R}$ of degree at most t, the average value of p on the set equals the average value of p on the whole sphere, that is,*

$$\frac{1}{N}\sum_{i=1}^{N} p(y_i) = \frac{1}{|S^d|}\int_{S^d} p(y)dw(y), \tag{1}$$

where $|S^d|$ is the surface of the whole unit sphere S^d and $dw(y)$ denotes the surface measure on S^d.

The concept of a spherical t-design was introduced by Delsarte, Goethals and Seidel [5] in 1977. The existence of a spherical t-design for any $t \geq 1$ and $d \geq 1$ was proved by Seymour and Zaslavsky in 1984 [20]. However, there is no known answer to the question of the number of points N needed to construct a spherical t-design for any $t \geq 1$ and $d \geq 1$.

For the case $d = 2$, finding spherical t-designs has many applications. The earth's surface is an approximate sphere S^2, and spherical t-designs are relevant to many problems of geophysics, including climate modeling and global navigation. Moreover, polynomial approximation on S^2 has wide applications in coding communications, viruses analysis and molecular chemistry. In this paper, we focus our study on spherical t-designs for S^2.

Let $\mathbb{P}_t$ be the linear space of restrictions of polynomials of degree $\leq t$ in 3 variables to S^2. Since the surface of the whole unit sphere S^2 is 4π, the equality (1) for S^2 can be written as

$$\int_{S^2} p(y)dw(y) = \frac{4\pi}{N}\sum_{i=1}^{N} p(y_i), \quad \text{for all } p \in \mathbb{P}_t. \tag{2}$$

A lower bound on the number of points N_t needed to construct a spherical t-design for any $t \geq 1$ and $d = 2$ was given in [5]:

$$N_t \geq N_t^* = \begin{cases} \frac{1}{4}(t+1)(t+3) & \text{if } t \text{ is odd} \\ \frac{1}{4}(t+2)^2 & \text{if } t \text{ is even.} \end{cases}$$

However, it is shown in [5] that the lower bound can not be achieved, that is, there is no spherical t-design with N_t^* points for any $t \geq 2$.

Hardin and Sloane [8] proposed a sequence of putative spherical t-designs with $\frac{1}{2}t^2 + o(t^2)$ points and presented numerical spherical t-designs for $t \leq 21$. Sloan and Womersley [22] established a new variational characterization of spherical designs: it is shown that a set $Y = \{y_1, \ldots, y_N\} \subset S^2$ is a spherical t-design if and only if a certain non-negative quantity vanishes. Using their characterization, Sloan and Womersley numerically obtained spherical t-designs for $t \leq 19$.

The dimension of the space $\mathbb{P}_t$ is $d_t := (t+1)^2$. A set of points $Y = \{y_1, \ldots, y_{d_t}\}$ is called a *fundamental system* if the zero polynomial is the only member of $\mathbb{P}_t$ that vanishes at each point $y_j, j = 1, \ldots, d_t$, which is equivalent to the $(t+1)^2 \times (t+1)^2$ Gram matrix at these points being nonsingular. Let $\mathcal{Y}$ denote the set of all fundamental systems. Chen and Womersley [4] presented a characterization of fundamental spherical t-designs: it is shown that a system $Y = \{y_1, \ldots, y_N\} \in \mathcal{Y}$ is a spherical t-design if and only if $Y \in \mathcal{Y}$ is a solution of a certain system of nonlinear equations. Chen and Womersley numerically computed approximate solutions for this system of nonlinear equations. They then numerically checked that the hypothesis of an appropriate generalization of the Newton–Kantorovich theorem holds. In this manner, they proved the existence of an exact solution of the nonlinear system close to the numerically computed approximation—and thus the existence of a t-design with d_t points. This approach is quite expensive computationally and was thus carried out for $t \leq 20$, only. Also, the hypothesis check for the Newton–Kantorovich type theorem was done using floating point arithmetic, so that numerical rounding prevents this approach from providing a proof in a strict mathematical sense.

It is known that finding spherical t-designs for large t is very difficult [2]. Up to now, spherical t-designs for $t \geq 21$ have not been verified. Evaluating large scale polynomials of high degree exactly or with known tight error bounds is a challenging problem in practical computation. In this paper, we use the characterization of fundamental spherical t-designs in [4] and interval arithmetic to find exact spherical t-designs with d_t points for t up to 100. Since also the effects of floating point rounding are completely accounted for, this computational approach has the quality of a "true" mathematical proof. In particular, we will compute tight bounds for a set of points $Y = \{y_1, \ldots, y_{(t+1)^2}\} \subset S^2$ which is proved to be a fundamental sys-

tem as well as a solution of the system of nonlinear equations from [4] which makes it a spherical t-design.

It is worth noting that just being a solution of the system of nonlinear equations is not sufficient to be a spherical t-design, see Example 1 below. So we really have to verify that the set of points is not only a solution of the system of nonlinear equations, but also a fundamental system.

In Section 2, we describe the characterization of fundamental spherical t-designs from [4] and our approach to find a solution of the system of nonlinear equations. In Section 3 we recall Krawczyk's method which uses interval arithmetic to provide narrow intervals for each component of a vector which provably contains a solution of the system of nonlinear equations. We also show how one can use interval arithmetic to prove the nonsingularity of the Gram matrix over this interval vector. This then shows that the solution contained in the interval vector is also a fundamental system and thus a t-design. Moreover, the radii of the intervals enclosing the components are at most $\sim 10^{-9}$ for t up to 100, so the coordinates of the points of the t-design are known to high accuracy.

In Section 4, we present our numerical results which prove the existence of t-designs with d_t points for t up to 100. So our work provides strong evidence that spherical t-designs with $(t+1)^2$ points will probably exist for any t. In this sense, this paper further contributes to the solution of the open problem on the minimal number of points needed to construct a spherical t-design. Moreover, our results suggest that for any $t \geq 1$, there is a *fundamental* spherical t-design.

2 Spherical t-designs with $(t+1)^2$ points

A set of points

$$Y = \{y_1, \ldots, y_{d_t}\} \subset S^2 \tag{3}$$

is termed an *extremal system* if its points maximize the determinant of the interpolation matrix with respect to an arbitrary basis of $\mathbb{P}_t$. Such extremal systems have been shown to have excellent geometrical properties since they tend to be evenly distributed over the sphere, and this is also the reason why they are well-conditioned for the purposes of numerical integration, see [21]. (Tables of numerically computed extremal sets can be found at Womersley's web site [24].) An extremal set with $(t+1)^2$ points is usually not a t-design. For this reason we aim at finding t-designs with $(t+1)^2$ points close to those of an extremal system.

Our starting point is the result from [4] which characterizes the $(t+1)^2$ points of a spherical t-design as the zeros of a nonlinear function $c : \mathbb{R}^{m_t} \to \mathbb{R}^{n_t}$, where $m_t = 2d_t - 3$ and $n_t = d_t - 1$. We need some preparations before we will be able to formulate this result precisely.

Definition 2 *The* Legendre *polynomials* $L_\ell(s), \ell = 0, 1, \ldots,$ *are defined through the recurrence*

$$\begin{aligned} L_0(s) &\equiv 1, \\ L_1(s) &= s, \\ \ell \cdot L_\ell(s) &= (2\ell - 1)s \cdot L_{\ell-1}(s) - (\ell - 1) \cdot L_{\ell-2}(s) \quad \textit{for } \ell = 2, 3, \ldots. \end{aligned}$$

The Jacobi *polynomials* $J_t(s)$ *are given as*

$$J_t(s) = \sum_{\ell=0}^{t} (2\ell + 1) L_\ell(s).$$

For a system Y of points as given in (3) we define the Gram matrix $G = G(Y)$ using the Jacobi polynomial J_t via

$$G_{ij} = J_t(y_i^T y_j), \quad i, j = 1, \ldots, d_t.$$

G is symmetric and it is known to be positive semidefinite, see [4]. If G is nonsingular, then the functions

$$g_i(y) = J_t(y_i^T y), \quad i = 1, \ldots, d_t,\ y \in S^2$$

form a basis for $\mathbb{P}_t$. This implies that the zero polynomial is the only member of $\mathbb{P}_t$ that vanishes at each point $y_i, i = 1, \ldots, d_t$. Hence, the set of points $\{y_1, \ldots, y_{d_t}\}$ is a fundamental system.

We represent the points $y_i \in S^2$ using polar coordinates with angles θ_i, φ_i. Since all spherical t-designs which can be mapped upon each other via a rotation on the sphere can be regarded to be equivalent, we can fix y_1 as being the north pole and y_2 as lying on the zero meridian, i.e. we have

$$y_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad y_2 = \begin{bmatrix} \sin(\theta_2) \\ 0 \\ \cos(\theta_2) \end{bmatrix}, \quad y_i = \begin{bmatrix} \sin(\theta_i)\cos(\varphi_i) \\ \sin(\theta_i)\sin(\varphi_i) \\ \cos(\theta_i) \end{bmatrix}, \quad i = 3, \ldots, d_t. \tag{4}$$

Putting

$$x_\theta = [\theta_2, \ldots, \theta_{d_t}]^T, \ x_\varphi = [\varphi_3, \ldots, \varphi_{d_t}]^T \text{ and } x = \begin{bmatrix} x_\theta \\ x_\varphi \end{bmatrix} \in \mathbb{R}^{m_t}, \tag{5}$$

we obtain a parameterization $Y(x) = \{y_1(x), \ldots, y_{d_t}(x)\}$. We finally define the function

$$c : \mathbb{R}^{m_t} \to \mathbb{R}^{n_t},\ x \mapsto E \cdot G(Y(x)) \cdot e, \tag{6}$$

where

$$E = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 1 & 0 & -1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \cdots & 0 \\ 1 & 0 & \cdots & 0 & -1 \end{bmatrix} \in \mathbb{R}^{n_t \times d_t},\ \ e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^{d_t}.$$

The following result was proved in [4, Theorem 3.1].

Theorem 1 *Let x be as in (5) and suppose that $G(Y(x))$ is nonsingular. Then x is the parameterization of a spherical t-design $Y(x)$ with $(t+1)^2$ points if and only if $c(x) = 0$.*

This theorem means that if x^* is a solution of $c(x) = 0$ and $G(Y(x^*))$ is nonsingular, then $Y(x^*)$ is a spherical t-design. The following example shows that nonsingularity of $G(Y(x^*))$ is not a necessary condition for $Y(x^*)$ being a spherical t-design, but if $G(Y(x^*))$ is singular, then $c(x^*) = 0$ does not ensure that $Y(x^*)$ is a spherical t-design.

Example 1 Take $t = 1$, so that $d_t = 4$. We fix y_1 as being the north pole and y_2 as lying on the zero meridian.

1. Choosing the four points

$$y_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},\ \ y_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},\ \ y_3 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix},\ \ y_4 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix},$$

the Gram matrix G at these four points $Y = \{y_1, y_2, y_3, y_4\}$ is

$$G(Y) = \begin{pmatrix} 4 & 1 & -2 & 1 \\ 1 & 4 & 1 & -2 \\ -2 & 1 & 4 & 1 \\ 1 & -2 & 1 & 4 \end{pmatrix}.$$

So $G(Y)$ is singular. Moreover, we have

$$G(Y)e = [4,\, 4,\, 4,\, 4]^T, \quad \text{and} \quad EG(Y)e = [0,\, 0,\, 0]^T.$$

Checking (1) for the four basis functions in $\mathbb{P}_1$,

$$p_1(y) = 1, \quad p_2(y) = \alpha, \quad p_3(y) = \beta, \quad p_4(y) = \gamma,$$

where $y = (\alpha, \beta, \gamma)^T$, we find

$$\frac{1}{4}\sum_{i=1}^{4} p_1(y_i) = \frac{1}{4\pi}\int_{S^2} p_1(y)dw(y) = 1,$$

and

$$\frac{1}{4}\sum_{i=1}^{4} p_j(y_i) = \frac{1}{4\pi}\int_{S^2} p_j(y)dy = 0, \quad j = 2, 3, 4.$$

Hence Y is a spherical t-design, although $G(Y)$ is singular.

2. Choosing the four points

$$y_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad y_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad y_3 = \frac{1}{2}\begin{bmatrix} 1 \\ -\sqrt{2} \\ 1 \end{bmatrix}, \quad y_4 = \frac{1}{2}\begin{bmatrix} 1 \\ \sqrt{2} \\ 1 \end{bmatrix},$$

the Gram matrix G at these points $Y = \{y_1, y_2, y_3, y_4\}$ is

$$G(Y) = \begin{pmatrix} 4 & 1 & 2.5 & 2.5 \\ 1 & 4 & 2.5 & 2.5 \\ 2.5 & 2.5 & 4 & 1 \\ 2.5 & 2.5 & 1 & 4 \end{pmatrix}$$

which is singular. Moreover, we have

$$G(Y)e = (10, 10, 10, 10)^T, \quad \text{and} \quad EG(Y)e = (0, 0, 0)^T.$$

Checking (1) for the four basis functions in $\mathbb{P}_1$ as in the first case, we find

$$\frac{1}{4}\sum_{i=1}^{4} p_4(y_i) = \frac{1}{2} \quad \text{but} \quad \frac{1}{4\pi}\int_{S^2} p_4(y)dw(y) = 0.$$

Hence Y is not a spherical t-design.

Theorem 1 sets the stage for the remaining part of this paper, where our goal is to computationally prove the existence of a zero of c close to an extremal system. The generic approach is as follows:

1. Take an extremal system as an initial guess for the Gauss–Newton method to solve $c(x) = 0$. The result of the Gauss–Newton method is an approximate zero $\hat{x}$ of c.
2. Starting from $\hat{x}$, construct a set $\mathcal{X}$ for which we can show that
   a) $\mathcal{X}$ contains a solution of $c(x) = 0$
   b) $G(Y(x))$ is non-singular for all $x \in \mathcal{X}$.
   By Theorem 1 we then know that $\mathcal{X}$ contains a parameterization x^* of a spherical t-design with $(t+1)^2$ points.

For Step 2, the following approach has been adopted in [4]: The components of $\hat{x}$ corresponding to the φ angles were considered fixed so that $c(x)$ is now regarded as a function c from $\mathbb{R}^{m_t}$ into itself depending only on the variables x_θ corresponding to the θ angles. The existence of a zero of c in a neighborhood of $\hat{x}$ is then proved by using a variant of the Newton–Kantorovich theorem developed in [4]. There, the neighborhoods are taken as balls in the ℓ_∞-norm, i.e. as interval vectors. For the theorem to be applicable, one needs a Lipschitz constant for the first derivative $c'(x_\theta)$ in the neighborhood which was obtained in [4] by bounding all entries of the second derivative using an interval arithmetic evaluation. Interval arithmetic was also used to obtain a bound for the norm of the first derivative in the neighborhood. Other required quantities such as $\|c'(\hat{x}_\theta)^{-1}c(\hat{x}_\theta)\|$ were computed in standard floating–point arithmetic. So, checking the hypothesis of the Newton–Kantorovich type theorem in this manner leaves a fundamental uncertainty because the effects of roundoff are not taken into account.

In the present paper we use a different approach. The idea is to use interval arithmetic and the Krawczyk operator to computationally prove the existence of a zero. We perform all computations in *machine interval arithmetic* so that rounding errors are completely taken into account due to outward rounding. Our computational existence proofs therefore are mathematically rigorous. As opposed to [4], we also adopt a more flexible strategy for deciding which variables are to be considered fixed. Finally, our approach tries to vectorize as many interval operations as possible, which, as we will see, is crucial for an efficient implementation. As a result, we are now able to computationally prove the existence of t-designs with $(t+1)^2$ points for t up to 100, whereas the approach from [4] was time critical already for $t = 20$. Let us note that both approaches have a time complexity of $\mathcal{O}(t^6)$.

3 Enclosing zeros of functions using interval arithmetic

We start by briefly introducing interval arithmetic and the notation we use. By $\mathbb{IR}$ we denote the space of all compact real intervals $\boldsymbol{a} = [\underline{a}, \overline{a}], \underline{a}, \overline{a} \in \mathbb{R}, \underline{a} \leq \overline{a}$. The arithmetic operations $+, -, *, /$ can be extended from $\mathbb{R}$ to $\mathbb{IR}$ in the usual set theoretic sense, and the bounds of the resulting intervals can be computed from the bounds of the operands. The real numbers can be embedded into the space of intervals by identifying each number $a \in \mathbb{R}$ with the "point interval" $\boldsymbol{a} = [a, a]$. We refer to [1] for details.

We try to keep to the standard notations of interval analysis defined in [9]. So all interval quantities will be typeset in boldface. For an interval $\boldsymbol{a} = \lfloor \underline{a}, \overline{a} \rceil \in \mathbb{IR}$ its diameter is $\operatorname{diam}(\boldsymbol{a}) = \overline{a} - \underline{a}$, its radius is $\operatorname{rad}(\boldsymbol{a}) = \operatorname{diam}(\boldsymbol{a})/2$, its midpoint is $\operatorname{mid}(\boldsymbol{a}) = (\overline{a} + \underline{a})/2$ and its absolute value is $|\boldsymbol{a}| := \max\{|a| \mid a \in \boldsymbol{a}\} = |\operatorname{mid}(\boldsymbol{a})| + \operatorname{rad}(\boldsymbol{a})$. The hull $\Box(\boldsymbol{a}, \boldsymbol{b})$ of two intervals in $\mathbb{IR}$ is the interval of smallest radius containing $\boldsymbol{a}$ and $\boldsymbol{b}$.

For interval vectors and matrices, $\operatorname{rad}, \operatorname{mid}, |\cdot|, \Box$ will be applied componentwise, thus producing results of the same dimension as the arguments.

Any arithmetic expression involving the components $x_1, \ldots, x_m$ of a vector $x \in \mathbb{R}^m$ defines a function $\varphi : D \subseteq \mathbb{R}^m \to \mathbb{R}$, and n such expressions give a function $f : D \subseteq \mathbb{R}^m \to \mathbb{R}^n$, one for each component f_i of f. By slight abuse of notation we identify the functions with their arithmetic expressions. Replacing the real vector x by an interval vector $\boldsymbol{x} \in \mathbb{IR}^m$ we thus obtain an interval extension $\boldsymbol{f}$ of f. Note that in contrast to the point case $\boldsymbol{f}$ will in general depend on the expression representing f. By the inclusion property of interval arithmetic, the range of f over an interval is contained in its interval extension, i.e.

$$\{f(x) : x \in \boldsymbol{x}\} \subseteq \boldsymbol{f}(\boldsymbol{x}). \tag{7}$$

The inclusion property is the key for obtaining computational proofs through the use of interval arithmetic. It is therefore of vital importance to implement interval arithmetic on a computer in such a manner that (7) always holds despite the occurrence of rounding errors. This is achieved in *machine interval arithmetic*, where the interval bounds are from the floating point screen and outward rounding is used to guarantee that the computed interval always contains the "true" result of an interval arithmetic operation. Two software systems which provide for such a machine interval arithmetic are the INTLAB toolbox [19] for MATLAB and the C++ class library C-XSC [10,11]. In order to obtain computational speed comparable to that of standard floating–point arithmetic, in INTLAB particular care has been taken to avoid excessive switching of rounding modes in standard interval matrix operations. For this reason, the default interval arithmetic of INTLAB is actually the restriction of complex circular arithmetic (see [1]) to real intervals, which for $*$ and / produces slightly larger intervals than in the set theoretic definition. The crucial inclusion property (7), however, does hold for INTLAB's machine interval arithmetic. We refer to [19] for a thorough discussion of INTLAB's other main features.

Now assume that f is continuously differentiable and let $x, y \in \boldsymbol{x}$. Then, by the mean value theorem we have

$$f_i(y) = f_i(x) + f_i'(\xi_i)(y - x),\ \xi_i = \alpha_i x + (1 - \alpha_i)y \text{ for some } \alpha_i \in [0, 1]$$

for each component $f_i, i = 1, \ldots, n$ of f. Let $\boldsymbol{A} \in \mathbb{IR}^{n \times m}$ be an interval matrix which contains $f'(\xi)$ for all $\xi \in \boldsymbol{x}$, i.e. $\boldsymbol{A}_{ij}$ contains $\partial f_i(\xi)/\partial x_j$ for all $\xi \in \boldsymbol{x}$. Such $\boldsymbol{A}$ may, for example, be obtained as an interval arithmetic evaluation of (expressions for) the gradients f_i' at the interval vector $\boldsymbol{x}$. The inclusion property of interval arithmetic gives

$$f(y) \in f(x) + \boldsymbol{A}(y - x) \tag{8}$$

and

$$\{f(y) : y \in \boldsymbol{x}\} \subseteq f(x) + \boldsymbol{A}(\boldsymbol{x} - x).$$

In what follows we now assume $n = m$. For a given matrix $C \in \mathbb{R}^{n \times n}$ and for $\check{x} \in \boldsymbol{x} \subset D$ and $\boldsymbol{A} \in \mathbb{IR}^{n \times n}$ the *Krawczyk operator* $\boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{A})$ is now defined as

$$\boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{A}) = \check{x} - Cf(\check{x}) + (I - C \cdot \boldsymbol{A})(\boldsymbol{x} - \check{x}). \tag{9}$$

The above discussion shows that $\boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{A})$ contains the range of the function $g(x) = x - Cf(x)$ over $\boldsymbol{x}$. If C is taken close to the inverse of $f'(\check{x})$, the function g will be contractive on $\boldsymbol{x}$ if $\boldsymbol{x}$ is sufficiently small. For this reason, the Krawczyk operator may be used for a computational existence test. The precise result is as follows:

Theorem 2 *Let $f : D \subset \mathbb{R}^n \to \mathbb{R}^n$ be as before, $\boldsymbol{x} \in \mathbb{IR}$ such that $\boldsymbol{x} \subset D$, $\check{x} \in \boldsymbol{x}$ and $\boldsymbol{A} \in \mathbb{IR}^{n \times n}$ such that $f'(\xi) \in \boldsymbol{A}$ for all $\xi \in \boldsymbol{x}$. Assume that*

$$\boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{A}) \subset \operatorname{int} \boldsymbol{x}, \tag{10}$$

where $\operatorname{int} \boldsymbol{x}$ is the topological interior of $\boldsymbol{x}$. Then f has a zero x^* in $\boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{A})$ which is unique in $\boldsymbol{x}$.

As stated, this theorem is due to Rump [18]. It goes back to Krawczyk [13], see also [14]. Note that since $\boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{A})$ contains the range of g over $\boldsymbol{x}$, condition (10) immediately implies that g has a fixed point by Brouwer's theorem. For this fixed point to be a zero of f we need that C is non-singular. This follows from (10), too, where now the fact that $\boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{A})$ is mapped into the *interior* of $\boldsymbol{x}$ is crucial, as it is for the uniqueness of x^*, see [18].

When we want to use Theorem 2 for a computational existence proof we have to be aware that $\boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{A})$ cannot be computed exactly due to floating point rounding errors. In order for the theorem to be applicable in machine computation, we therefore have to make sure

that the computed quantity $\tilde{\boldsymbol{k}}$ *contains* $\boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{A})$. If we then have $\tilde{\boldsymbol{k}} \subset \operatorname{int}(\boldsymbol{x})$ we also have (10). One way to achieve $\tilde{\boldsymbol{k}} \supseteq \boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{A})$ is to carry out *all* operations as (machine arithmetic) interval operations, which also means that we use point intervals instead of real quantities throughout. In this manner, rounding errors which would occur in floating point arithmetic will be accounted for by the machine interval arithmetic result which contains the exact result. This observation applies in particular to the evaluation of $f(\check{x})$, which we expect to be close to zero. If f is represented by an arithmetic expression involving many operations—as is the case for the function c from (6)—, the many outward roundings will produce an interval vector $\boldsymbol{f}$ with relatively large intervals enclosing the components of $f(\check{x})$. Since $\operatorname{rad}(C\boldsymbol{f}) \leq \operatorname{rad}(\tilde{\boldsymbol{k}})$ and because $\tilde{\boldsymbol{k}} \subset \operatorname{int}(\boldsymbol{x})$ can only hold if $\operatorname{rad}(\tilde{\boldsymbol{k}}) < \operatorname{rad}(\boldsymbol{x})$, we see that $\operatorname{rad}(C\boldsymbol{f})$ is a limiting bound for $\operatorname{rad}(\boldsymbol{x})$ if we want (10) to hold, i.e. for the accuracy of the guaranteed enclosures that we will be able to obtain. Since $\operatorname{rad}(C\boldsymbol{f}) = |C| \cdot \operatorname{rad}(\boldsymbol{f})$ (see [1], e.g.) we see that narrow enclosures can be obtained only if the two requirements

$$|C| \text{ is small} \quad \text{and} \tag{11}$$

$$\operatorname{rad}(\boldsymbol{f}) \text{ is small} \tag{12}$$

are both met. The next two subsections show how we try to achieve both these goals in the context of the verification of t-designs. We thus turn back to the function $c : \mathbb{R}^{m_t} \to \mathbb{R}^{n_t}$ from (6), for which an approximate zero $\hat{x}$ is assumed to be available.

3.1 Choosing the independent variables

Let $\{1, \ldots, m_t\} = \mathcal{B} \cup \mathcal{N}$ be a partitioning such that $\mathcal{B}$ has size n_t and let us write $x = (x_{\mathcal{B}}, x_{\mathcal{N}})$ to denote the induced partitioning of the vector $x \in \mathbb{R}^{m_t}$ into two blocks, and similarly for $\hat{x} = (\hat{x}_{\mathcal{B}}, \hat{x}_{\mathcal{N}})$. Putting

$$c_{\mathcal{B}}(x_{\mathcal{B}}) = c((x_{\mathcal{B}}, \hat{x}_{\mathcal{N}}))$$

we obtain a function $c_{\mathcal{B}} : \mathbb{R}^{n_t} \to \mathbb{R}^{n_t}$ which represents the function c with the variables corresponding to $\mathcal{N}$ kept fixed. Since we want to use the Krawczyk operator for $c_{\mathcal{B}}$, in the light of requirement (11) the index set $\mathcal{B}$ should be chosen such that $c'_{\mathcal{B}}(\hat{x}_{\mathcal{B}})$ has an inverse which is small. Note that the columns of $c'_{\mathcal{B}}$ are a subset of the columns of c', so the task is to choose the right set of columns. To this purpose, we use a column pivoting QR-decomposition of $c'(\hat{x})$, see [7]. It yields a decomposition

$$c'(\hat{x}) \cdot P = Q\,[\; R \mid * \;], \quad P \in \mathbb{R}^{m_t \times m_t},\, Q, R \in \mathbb{R}^{n_t \times n_t},$$

where P is a permutation matrix, Q is orthogonal and R is upper triangular. The permutation P arises from a greedy strategy to obtain maximum diagonal elements in R. Taking $\mathcal{B}$ as those components which are permuted to the first n_t positions by P, we get

$$c'_{\mathcal{B}}(\hat{x}_{\mathcal{B}})^{-1} = R^{-1}Q^T.$$

We expect $c'_{\mathcal{B}}(\hat{x}_{\mathcal{B}})^{-1}$ to be small since Q^T is orthogonal and R^{-1} is the inverse of a triangular matrix with large diagonal entries.

3.2 Computing narrow enclosures for the function values

As can be seen from (6), tight enclosures for a function value $c_{\mathcal{B}}(\hat{x}_{\mathcal{B}}) = c(\hat{x})$ are achievable only if one is able to compute tight enclosures for all the entries of $G(Y(\hat{x}))$, i.e. for $J_t(y_i^T y_j)$, $i, j = 1, \ldots, d_t$. Since we have to control rounding errors in every stage, we have to perform all operations as machine interval arithmetic operations. To this end we identify the components of $\hat{x}$ (the angles θ_i and φ_i) with degenerate intervals $\boldsymbol{\theta}_i$ and $\boldsymbol{\varphi}_i$, resp., containing just one element, and from these we compute enclosures $\boldsymbol{s}_{ij}$ for the scalar products $s_{ij} = y_i(\hat{x})^T y_j(\hat{x})$, which will be the arguments of J_t later on.

Since the width of $J_t(\boldsymbol{s})$ is roughly linear in the width of the argument, it is essential that the enclosures $\boldsymbol{s}_{ij}$ be narrow. To achieve this, we use four different equivalent formulae for the (point) quantities s_{ij}.

- According to the definition $s_{ij} = y_i(\hat{x})^T y_j(\hat{x})$ and the representations (4), we first compute the interval vectors

$$\boldsymbol{y}_i = (\sin(\boldsymbol{\theta}_i)\cos(\boldsymbol{\varphi}_i), \sin(\boldsymbol{\theta}_i)\sin(\boldsymbol{\varphi}_i), \cos(\boldsymbol{\theta}_i))^T,$$

  and from these the enclosures $\boldsymbol{s}_{ij}^{(1)} = \boldsymbol{y}_i^T \cdot \boldsymbol{y}_j$.
- Spherical geometry tells us that s_{ij}, which is the cosine of the angle between the normalized vectors $y_i(\hat{x})$ and $y_j(\hat{x})$, is given by $s_{ij} = \cos(\theta_i)\cos(\theta_j) + \sin(\theta_i)\sin(\theta_j)\cos(\varphi_i - \varphi_j)$. Evaluating this formula for the point intervals $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ leads to an enclosure $\boldsymbol{s}_{ij}^{(2)}$.
- Geometric identities yield $s_{ij} = \cos(\theta_i)\cos(\theta_j)(1 - \cos(\varphi_i - \varphi_j)) + \cos(\theta_i - \theta_j)\cos(\varphi_i - \varphi_j) = \cos(\theta_i - \theta_j) + \sin(\theta_i)\sin(\theta_j)(\cos(\varphi_i - \varphi_j) - 1)$, which gives two more enclosures $\boldsymbol{s}_{ij}^{(3)}$ and $\boldsymbol{s}_{ij}^{(4)}$.

Note that for these computations we need interval versions of the trigonometric functions as they are available in INTLAB, for example. By taking $\boldsymbol{s}_{ij}$ to be the intersection of the four enclosures $\boldsymbol{s}_{ij}^{(1)}$, $\ldots$, $\boldsymbol{s}_{ij}^{(4)}$, the diameter of the $\boldsymbol{s}_{ij}$ is reduced by about one half (to

$\leq 10\epsilon_{\text{mach}}$, where $\epsilon_{\text{mach}} = 2^{-53} \approx 2.2 \cdot 10^{-16}$ denotes the machine epsilon), as compared to the straight-forward enclosure $\boldsymbol{s}_{ij}^{(1)}$. This improvement is paid for by a substantial increase in computing time. In fact, computing the arguments $\boldsymbol{s}_{ij}$ for J_t in this manner accounts for approximately one half of the overall time for evaluating the function $c(\hat{x})$.

The next task is to obtain *tight* enclosures for the range of *the same polynomial* J_t over *a large number* of (narrow) intervals $\boldsymbol{s}_{ij}$, all lying in $[-1, 1]$. A first study for various approaches was outlined by two of the authors in [6]. We now present the relevant points in detail and illustrate them with many numerical results.

The polynomial J_t is evaluated over the $d_t^2 = (t+1)^4$ intervals $\boldsymbol{s}_{ij}$, $i, j = 1, \ldots, d_t$. (Exploiting the symmetry $\boldsymbol{s}_{ij} = \boldsymbol{s}_{ji}$, the number of arguments might be reduced to $d_t(d_t + 1)/2$, but we did not do this in our implementation.)

An interval arithmetic based method for these many computations is significantly more efficient if it can be *vectorized*, i.e. if the individual computational steps can be performed simultaneously, since then we avoid many costly switchings of the rounding mode. On the other hand, there is also the issue of the tightness of the enclosures that we will compute. It turns out that methods based on recurrences similar to that of Definition 2, while vectorizable, yield enclosing intervals which prohibitively overestimate the exact range. For example, for $t = 40$ and $\boldsymbol{s}$ an interval close to 1 with diameter $10\epsilon_{\text{mach}}$, these methods yield intervals with a radius larger than 1, while the exact range has diameter $\approx 10^{-8}$ only. The same excessive over estimation arises if we pre-calculate all the coefficients γ_j in the expansion

$$J_t(s) = \sum_{j=0}^{t} \gamma_j s^j \tag{13}$$

and then use the Horner scheme to evaluate $J_t(\boldsymbol{s})$ using interval arithmetic. It is possible to treat the recurrences as results of a parameter dependent linear system for which specialized interval methods as presented in [12,15,17] are available. This yields tight enclosures, but now the methods do not vectorize and so using INTLAB the computational cost becomes more than 100 times higher. Exactly the same holds if one treats Horner's scheme as a linear system as suggested in [3], e.g.

The most satisfying approach with respect to quality of the enclosures *and* computational efficiency is a *multi–point* Horner scheme. The idea is to first set up a preparatory stage where we choose a set

of $k+1$ points σ_i, $i = 0, \ldots, k$, in the interval $[-1, 1]$ and compute the coefficients γ_{ij} of each Taylor expansion of J_t centered at any of the points σ_i,

$$J_t(s) = \sum_{j=0}^{t} \gamma_{ij}(s - \sigma_i)^j. \tag{14}$$

Taking $k = 2^q$ and $\sigma_i = -1 + i \cdot 2^{-q+1}, i = 0, \ldots, k$, the points σ_i have an exact representation as floating point numbers. Since the coefficients in Definition 2 for the Legendre and Jacobi polynomials are integers, we use rational arithmetic to compute the coefficients γ_{ij} exactly. To do this efficiently, we wrote the code for this part in C. Enclosures for the γ_{ij} into narrow intervals are written to a file that is read by our INTLAB code which performs all the remaining parts of the computation.

Now, if we have to compute the enclosure of the range of J_t over an interval $\boldsymbol{s}$, we choose the two Taylor expansions corresponding to the points σ_i and σ_{i+1} with $\sigma_i \le \mathrm{mid}\,(\boldsymbol{s}) \le \sigma_{i+1}$, evaluate these two expansions in interval arithmetic using Horner's scheme, and finally take the intersection of the two resulting intervals as the enclosure for the range of J_t over $\boldsymbol{s}$. An additional benefit occurs in this approach if we are to compute enclosures for the range of J_t using a given expansion with center σ_i for a whole set of intervals $\boldsymbol{s}_\ell$, $\ell = 1, \ldots, k_i$, with small radii. Putting

$$\rho_i = \max_{\ell=1}^{k_i} |\boldsymbol{s}_\ell - \sigma_i|$$

we have

$$|(\boldsymbol{s}_\ell - \sigma_i)^j| \le \rho_i^j,$$

and ρ_i^j will be very small for j large. We can therefore precompute an "effective" degree $\delta_i(t)$ defined as

$$\delta_i(t) = \min \left\{ \nu : \sum_{\mu=\nu+1}^{t} |\gamma_{i\mu}| \cdot \rho^\mu \le \epsilon_{mach} \right\}.$$

So for $\ell = 1, \ldots, k_i$ we get an enclosure for the range of J_t over $\boldsymbol{s}_\ell$ by evaluating the truncated expansion

$$\sum_{j=0}^{\delta_i} \gamma_{ij}(\boldsymbol{s}_\ell - \sigma_i)^j + [-\epsilon_{\mathrm{mach}}, \epsilon_{\mathrm{mach}}]$$

via Horner's scheme. The reduction of the degree can be substantial, as illustrated in Figure 1, where for $q = 9$ and different t we report the

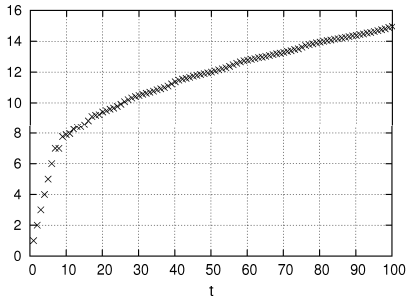

Fig. 1. Effective degree for the truncated multi–point Horner scheme.

effective degree of the truncated Taylor expansions averaged over all expansion points. This effective degree seems to grow logarithmically in t, only. It is still less than 15 for $t = 100$.

The choice $q = 9$, i.e., $k = 512$ turns out to be sufficient for the range $t = 1, \ldots, 100$, meaning that the radii of all thus computed enclosures are satisfactorily small. Indeed, for any interval $\boldsymbol{s}$ with radius of $10\epsilon_{\text{mach}}$, the computed enclosure has a radius close to $J'(\text{mid}\,(\boldsymbol{s}))\cdot\text{rad}\,(\boldsymbol{s})$ which is the best that we can expect, since the radius of the exact range of J_t over $\boldsymbol{s}$ will have a radius of approximately this size. The left part of Figure 2 illustrates the quality of the enclosures for $t = 40$ for various approaches. There, we give the diameter of the computed enclosure for the range of J_t over a series of intervals with midpoints varying from -1 to 1 and radius $10\epsilon_{\text{mach}}$. We compare the "standard method" which evaluates J_t using the recurrences from Definition 2, the plain Horner scheme where we use the Taylor expansion around $\sigma = 0$ for all interval arguments, the multi–point Horner scheme and its truncated variant just described as well as three variants of a method based on the expansion of J_t in terms of Chebyshev polynomials; see [6] for details. We also include the results obtained with the parameterized system approach from [12,15]. Very clearly, the multi–point Horner schemes and the parameterized system approach produce the best results, since the diameters of the results are uniformly small for the whole range of interval arguments. Two of the Chebyshev polynomial based approaches, `cheb1` and `cheb2`, which both use a log-depth recurrence, also yield uniform diameters, but these are 10 to 100 times larger. All other methods suffer severely from increasing overestimations as the interval argument approaches the boundaries -1 and $+1$.

The right part of Figure 2 compares the timings for the various approaches as a function of the number of intervals for which an enclosure for the range of J_t is needed. These intervals were generated

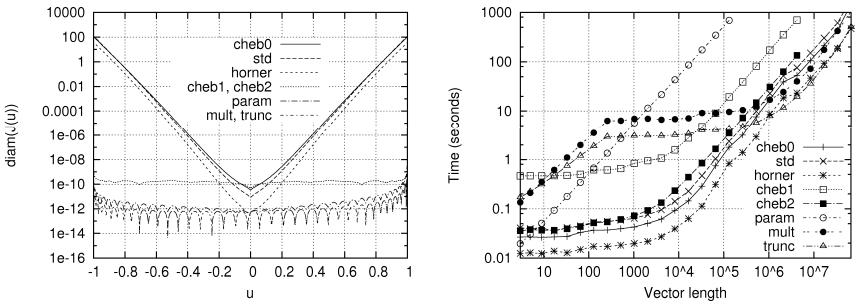

Fig. 2. Quality of enclosures for the range of J_t over intervals with radius $10\epsilon_{\text{mach}}$ (left), timings as a function of the number of evaluations (right). Both plots use $t = 40$.

randomly in $[-1, 1]$ as the components of a vector whose length is the abscissa of the plot. We again took $t = 40$, so that in order to get an enclosure for $c(x)$ we actually have to evaluate J_t at $41^4 \approx 2.8 \times 10^6$ intervals, i.e., the vector length is $2.8 \cdot 10^6$. The approaches discussed above were implemented in INTLAB and run on a Sun Fire X4140 x64 (two 2.3 GHz quad-core Opterons) with 8 GB main memory and 32 GB of virtual memory (swap space on the hard disk). We used only one processor of the machine and no explicit multithreading. The right part of Figure 2 drastically shows the benefits of vectorization in INTLAB which is possible for all methods except the parameterized system approach. For many algorithms the execution time is nearly constant over a whole range of (small) vector lengths, because it is completely dominated by the cost for switching rounding modes. As the vector length gets larger, the cost for the (vectorized) arithmetic operations dominates so that we observe an almost linear increase in time for all methods for $t \geq 10^6$. The multi–point Horner scheme and its truncated variant exhibit an interesting plateau when the vector length is in the range $200, \ldots, 2{\cdot}10^5$. The reason is again vectorization: The multi–point scheme uses 513 different Taylor expansions. So, on the average, the vector length for each evaluation has to be divided by 256 (recall that each interval goes into two expansions). In the given range, therefore, this "local" vector length is not yet large enough to balance the cost for the switching of the rounding mode.

The bottom line of our discussion is that the truncated multi–point Horner scheme is the method of choice. It yields the tightest enclosures *and* the least overall computational cost for those vector lengths $(t + 1)^4$ that we will be using. We also note that the two log-depth recurrence Chebyshev polynomial approaches `cheb1` and `cheb2` need significantly more storage, namely $t + 1$ times the vector

length, than the other methods where storage is just a few vectors, independently of t. The $\mathcal{O}(t^5)$ memory demands make these two approaches impractical for $t \geq 60$ (61 vectors of $61^4 \approx 13.8 \cdot 10^6$ 16-byte intervals lead to massive paging and times far beyond one hour) and completely infeasible for $t \geq 74$.

3.3 Computing enclosures for the derivative

In order to apply the Krawczyk operator to the function $c_{\mathcal{B}}$ we also must compute an interval matrix $\boldsymbol{A}$ containing all values of the derivative $c'_{\mathcal{B}}(x)$ for $x \in \boldsymbol{x}$. Herein, $\operatorname{rad}(\boldsymbol{x})$ will be small, since it represents the accuracy of the sought-after enclosure. Typical values are in the range $10^2\epsilon_{\text{mach}} - 10^5\epsilon_{\text{mach}}$. An explicit expression for the components of the derivative c' of c is given by

$$\frac{\partial c_i(x)}{\partial x_k} = \sum_{j=1}^{d_t} \left(J'_t(y_1^T y_j) \frac{\partial (y_1^T y_j)}{\partial x_k} - J'_t(y_{i+1}^T y_j) \frac{\partial (y_{i+1}^T y_j)}{\partial x_k} \right). \qquad (15)$$

Note that herein the partial derivatives $\frac{\partial (y_1^T y_j)}{\partial x_k}$ and $\frac{\partial (y_{i+1}^T y_j)}{\partial x_k}$ are non-zero only if k is from $\mathcal{B}$ and, in addition, x_k corresponds to one of the angles θ_j, φ_j, θ_{i+1} or φ_{i+1}. So each sum contains at most three non-vanishing terms. In order to obtain tight enclosures for the range of $\frac{\partial c_i(x)}{\partial x_k}$ over an interval vector $\boldsymbol{x}$ it is again crucial to get tight enclosures for the range of a polynomial, namely for the derivative J'_t of J_t. We therefore apply the same multi–point Horner scheme as discussed in Section 3.2 to obtain quite tight enclosures for all the ranges of J'_t over intervals $\boldsymbol{s}_{1j}$ and $\boldsymbol{s}_{i+1,j}$ which enclose all scalar products $y_1^T y_j$ and $y_{i+1}^T y_j$, resp., over a parameter range $\boldsymbol{x}$. In contrast to the evaluation of c, however, $\boldsymbol{x}$ is not a degenerate interval vector but has significant diameter. Therefore the computation of $\boldsymbol{s}_{ij} = \boldsymbol{s}_{ij}^{(1)}$ according to the first of the methods discussed at the beginning of Section 3.2 is sufficient. The evaluation of the polynomials can again be vectorized, so that the computation is cost efficient in INTLAB since the number of switchings of the rounding mode remains small.

3.4 Nonsingularity of the Gramian

Once we have obtained an enclosure $\boldsymbol{x}$ for a zero x^* of c, we must check whether $G(x^*)$ is non-singular in order to be sure that x^* represents a spherical t-design, see Theorem 1. Since we only know that x^* is contained in $\boldsymbol{x}$ without knowing x^* exactly, we have to show that

$G(x)$ is nonsingular for all $x \in \boldsymbol{x}$. This can be done computationally using the following lemma. Therein, we use the notation $\|\boldsymbol{B}\|_\infty$ for an interval matrix $\boldsymbol{B} \in \mathbb{IR}^{n\times n}$ to denote the quantity

$$\max_{i=1}^{n} \sum_{j=1}^{n} |\boldsymbol{B}_{ij}|.$$

Lemma 1 *Let $\boldsymbol{G} \in \mathbb{IR}^{n\times n}$ be an interval matrix and let $H \in \mathbb{R}^{n\times n}$. Then, if*

$$\|I - H\boldsymbol{G}\|_\infty < 1, \tag{16}$$

H as well as all matrices $G \in \boldsymbol{G}$ are non-singular.

The proof is trivial since by the inclusion property, $\|I-H\boldsymbol{G}\|_\infty < 1$ implies $\|I-HG\|_\infty < 1$ for all $G \in \boldsymbol{G}$. So the spectral radius ρ satisfies $\rho(I - HG) < 1$ which implies that 0 cannot be an eigenvalue of HG. So neither H nor G are singular.

In an actual computation, we will take H to be a computed approximate inverse of $\mathrm{mid}\,(\boldsymbol{G})$ and we perform all operations in machine interval arithmetic, thus using outward rounding. We will get an interval $\boldsymbol{r} = [\underline{r}, \overline{r}]$ enclosing $\|I - H\boldsymbol{G}\|_\infty$, and if $\overline{r} < 1$ we have computationally proved that (16) holds. Of course we again use the multi–point Horner scheme to get tight interval enclosures $\boldsymbol{G}_{ij}$ for the range of all entries $G_{ij} = J_t\left(y_i^T(x)y_j(x)\right)$ of G over $x \in \boldsymbol{x}$.

4 Numerical results

We summarize the discussion of Section 3 as Algorithm 1, in this manner giving a high level description of our computational verification and enclosure method for spherical t-designs with $(t+1)^2$ points. Therein, line 8 initializes the candidate interval vector $\boldsymbol{x}$ using the ϵ-inflation principle from [16,17]. Recall that the interval hull operator $\sqsupset$ is to be understood componentwise and that $\sqsupset(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{c}$ with $\boldsymbol{c}$ the smallest interval containing $\boldsymbol{a}$ and $\boldsymbol{b}$. Usually, ϵ-inflation yields a narrow interval vector $\boldsymbol{x}$ for which $\boldsymbol{k}_f(\check{x}, \boldsymbol{x}, \boldsymbol{f}'(\boldsymbol{x})) \subset \mathrm{int}\,(\boldsymbol{x})$ is quite likely to hold. Indeed, in all our computations this was always the case. The algorithm never arrived at line 19 so that the loop generated by this line was actually never executed.

Let us formally explore the complexity of Algorithm 1.

Lemma 2 *The time complexity of Algorithm 1 is $\mathcal{O}(t^6)$.*

Algorithm 1 Verification and enclosure algorithm for t-design with $(t+1)^2$ points

1: **Input:** $t \in \mathbb{N}$, extremal system $\tilde{Y} = \{\tilde{y}_1, \ldots, \tilde{y}_{d_t}\} \subset S^2$
2: **Output:** Intervals $\boldsymbol{\theta}_i, \boldsymbol{\varphi}_i$, $i = 1, \ldots, d_t$, such that for all i there exist $\theta_i \in \boldsymbol{\theta}_i, \varphi_i \in \boldsymbol{\varphi}_i$ such that the points $\{y(\theta_i, \varphi_i) : i = 1, \ldots, d_t\}$ are a spherical t-design
3: Rotate points $\tilde{y}_i$ of extremal system such that first is on north pole and second on zero meridian; then convert all points to angles θ_i and φ_i
4: Put $x = (\theta_2, \ldots, \theta_{d_t}, \varphi_3, \ldots, \varphi_{d_t})$ and perform Gauss–Newton method for $c(x)$ with initial guess $x = (\theta_2, \ldots, \theta_{d_t}, \varphi_3, \ldots, \varphi_{d_t})$ until convergence; result is $\hat{x}$ with $c(\hat{x}) \approx 0$.
5: Compute rank revealing QR decomposition of $c'(\hat{x})$ and determine set $\mathcal{B}$ for independent variables
6: Compute C as (approximate) inverse of $c'_{\mathcal{B}}(\hat{x}_{\mathcal{B}})^{-1}$ in floating point
7: Compute $\boldsymbol{f} \supseteq C c_{\mathcal{B}}(\hat{x}_{\mathcal{B}})$ in machine interval arithmetic using multi–point Horner
8: Initialize $\boldsymbol{x}_{\mathcal{B}} := \hat{x}_{\mathcal{B}} + \square(0, \boldsymbol{y}_{\mathcal{B}})$ where $\boldsymbol{y}_{\mathcal{B}} = \boldsymbol{f} + \boldsymbol{d}$, $\operatorname{mid}(\boldsymbol{d}) = 0, \operatorname{rad}(\boldsymbol{d}) = 0.1 \cdot \operatorname{rad}(\boldsymbol{f})$
9: Compute interval matrix $\boldsymbol{A}$ containing $\{c'_{\mathcal{B}}(x) : x \in \boldsymbol{x}_{\mathcal{B}}\}$ using multi–point Horner
10: **if** $\boldsymbol{k} := \boldsymbol{k}_{c_{\mathcal{B}}}(\hat{x}_{\mathcal{B}}, \boldsymbol{x}_{\mathcal{B}}, \boldsymbol{A}) \subset \operatorname{int} \boldsymbol{x}_{\mathcal{B}}$ **then**
11: Compute interval Gram matrix $\boldsymbol{G} = (J_t(\boldsymbol{y}_i(\boldsymbol{x})^T \boldsymbol{y}_j(\boldsymbol{x}))_{ij} \in \mathbb{IR}^{d_t \times d_t}$ using multi–point Horner
12: Compute approximate inverse H of $\operatorname{mid}(\boldsymbol{G})$ in floating point
13: **if** $\|I - H\boldsymbol{G}\|_\infty < 1$ **then**
14: STOP, components of $\boldsymbol{k}$ are the required enclosures for the angles from $\mathcal{B}$, the other angles are *exactly* those from $\hat{x}$
15: **else**
16: STOP, method failed because non-singularity of $G(x^*)$ could not be verified
17: **end if**
18: **else**
19: Increase $\operatorname{rad}(\boldsymbol{x}_{\mathcal{B}})$ by a factor of 2 and go to line 9
20: **end if**

Proof The Gauss–Newton algorithm for an underdetermined system with $\mathcal{O}(t^2)$ equations and $\mathcal{O}(t^2)$ variables needs $\mathcal{O}(t^6)$ operations per step. Assuming the number of steps to be bounded, the overall cost for line 4 is thus $\mathcal{O}(t^6)$. The rank-revealing QR factorization of a matrix with both dimensions being $\mathcal{O}(t^2)$ is an $\mathcal{O}(t^6)$ process, as is the computation of the inverses of R and $\operatorname{mid}(\boldsymbol{G})$ and of the matrix–matrix products $C\boldsymbol{A}$ (in the Krawczyk operator) and $H\boldsymbol{G}$. Evaluating J_t or J'_t on a vector using Horner's scheme for a given (truncated) expansion requires $\mathcal{O}(t)$ vector operations, so that the computation of all the $\mathcal{O}(t^4)$ entries of the Gram matrix is an $\mathcal{O}(t^5)$ process. From

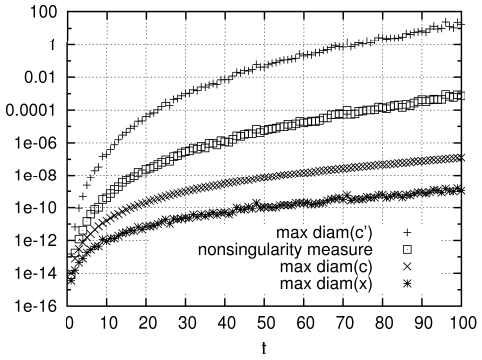

Fig. 3. Accuracy in final result $\boldsymbol{x}$, nonsingularity measure $\|I - H\boldsymbol{G}\|_\infty$, and diameters of the intermediate quantities $\boldsymbol{c}$ and $\boldsymbol{c}$'.

the definition of c and from (15) we thus see that evaluating c or c' has cost $\mathcal{O}(t^5)$.

We also have to address the cost for precomputing all the coefficients for the expansions (14) of the Jacobi polynomial and its derivative. For $k+1$ expansion points, these computations require $\mathcal{O}(k \cdot t^2)$ *rational* operations. The cost for each such operation cannot be quantified easily as the length of the numerators and denominators in intermediate results increases with t. The computation and saving of all the coefficients on a 2.13 GHz Pentium M laptop took 0.284/0.497/1.660/10.088/14.357 seconds for $t = 10/20/40/80/100$, which confirms the quadratic scaling in t and is well below one percent of the respective times for Algorithm 1; see below.

Algorithm 1 was implemented in INTLAB and run on the Sun Fire described in Section 3.2. On this machine we were able to prove the existence of spherical t-designs with $(t+1)^2$ points and to get narrow enclosures for the angles belonging to the points of such a design for $t = 1, \ldots, 100$. Presently, $t = 100$ represents the practical limit for our computational experiments, because Algorithm 1 uses much of the virtual memory and heavy paging sets in for larger values of t.

In Figure 3 we report, for each t, the maximum diameter of various interval quantities computed in Algorithm 1. Most importantly, `max diam(x)` represents the maximum diameter of all computed enclosures for the parameterization of the respective t-design. It thus represents the (guaranteed) accuracy to which we computed the parameterization. We see that this accuracy decreases as t increases, but it is still about 10^{-9} for the largest values of t. We also see that the nonsingularity measure $\|I - H\boldsymbol{G}\|_\infty$ from (16) is far away from its

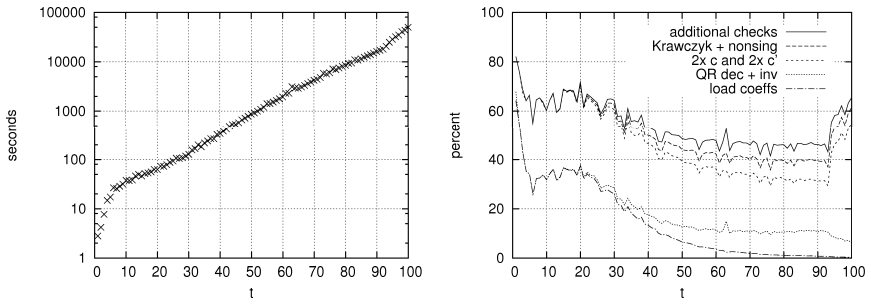

Fig. 4. Timings of the overall algorithm (left), and fraction of time spent in various computations from the verification part (right).

critical value 1 over the whole range for t. The diameters of the computed enclosures for the components of $c(\hat{x})$ (`max diam(c)`) are two orders of magnitude larger than those for the enclosure of the parameterization, thus indicating that the computed approximate inverse C is small. The diameters reported as `max diam(c')` refer to the components of $c'(\boldsymbol{x}_{\mathcal{B}})$. They are much larger than those for the components of c because we now evaluate at "true" intervals and because c' varies more rapidly than c.

The left part of Figure 4 reports the total execution time for various values of t. This includes the loading of the coefficients in the 513 expansions of the Jacobi polynomial and its derivative from file into INTLAB. For $t = 100$ the total time is about $50,000$ seconds, which is slightly more than half a day. Note that for $t \leq 93$ this diagram does not yet exhibit the $\mathcal{O}(t^6)$ dependence predicted by Lemma 2, indicating that for the range of t considered the computations with a complexity of $\mathcal{O}(t^5)$ or lower (such as an evaluation of the Gram matrix) are still predominant. The faster increase of the overall time for $t \geq 94$ is *not* due to the $\mathcal{O}(t^6)$ processes becoming dominant, but to the effects of paging because of insufficient main memory; see also the discussion below.

The right part of Figure 4 gives details of how much of the total execution time of Algorithm 1 is spent in the various steps which make up the computational *verification* process. The diagram is to be read cumulatively, so the uppermost line represents the overall time for all activities in the verification part. The important message is that the cost between the approximation part (basically the Gauss–Newton method performed at the beginning) and the verification part is well balanced and seems even to decrease in favor of the verification part as t increases. The most time consuming steps appear to be the QR factorization (a real arithmetic $\mathcal{O}(t^6)$ process)

and the evaluation of the Krawczyk operator including the computation of the nonsingularity measure (an interval arithmetic $\mathcal{O}(t^6)$ process), which both tend to take approximately the same time. As expected, the contribution of the lower-order computations (in each run we needed two evaluations of c and two evaluations of c') is decreasing for large t, up to $t = 93$. By contrast, these computations become dominant again for $t \geq 94$, which clearly indicates the setting in of paging, as the memory accesses are by far less localized in the evaluations than in the matrix–matrix operations. The exchange of data from the C-program to INTLAB, given as the lowermost curve, is negligible for all larger values of t.

5 Conclusions

This paper added to the theory of spherical t-designs by proving the existence of t-designs with $d_t = (t+1)^2$ points for t up to 100. The technique of proof is computational but rigorous through the use of machine interval arithmetic. One crucial ingredient to the success of the underlying Krawczyk method is the efficient and accurate evaluation of the Jacobi polynomials and their derivatives which we achieved using the truncated multi–point Horner scheme. Another important ingredient is the interval technique used to prove the nonsingularity of the Gram matrix. The data containing the enclosures for the parameterization of the t-designs and the programs are available from the web site `www-ai.math.uni-wuppertal.de/SciComp/SphericalTDesigns`.

References

1. G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Computer Science and Applied Mathematrics. Academic Press, New York, 1983.
2. E. Bannai and E. Bannai. A survey on spherical designs and algebraic combinatorics on spheres. *European Journal of Combinatorics*, to appear.
3. H. Böhm. Evaluation of arithmetic expressions with maximum accuracy. In Ulrich Kulisch and Willard M.Miranker, editors, *A New Approach to Scientific Computing*, pages 121–137. Academic Press, New York, 1983.
4. Xiaojun Chen and Robert S. Womersley. Existence of solutions to systems of underdetermined equations and spherical designs. *SIAM J. Numer. Anal.*, 44:2326–2341, 2006.
5. P. Delsarte, J. M. Goethals, and J. J. Seidel. Spherical codes and designs. *Geom. Dedicata*, 6:363–388, 1977.
6. Andreas Frommer and Bruno Lang. Fast and accurate multi-argument interval evaluation of polynomials. In *Proceedings of the 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and*

Validated Numerics (SCAN 2006), page 31, Los Alamitos, CA, USA, 2006. IEEE Computer Society.
7. Gene H. Golub and Charles F. Van Loan. *Matrix computations*. The Johns Hopkins Univ. Press, Baltimore, 3rd edition, 1996.
8. R. H. Hardin and N. J. A. Sloane. McLaren's improved snub cube and other new spherical designs in three dimensions. *Discrete Comput. Geom.*, 15:429–441, 1996.
9. R. B. Kearfott, M.T. Nakao, A. Neumaier, S.M. Rump, S.P. Shary, and P. van Hentenryck. Standardized notation in interval analysis, 2005. `http://www.mat.univie.ac.at/~neum/ms/notation.pdf`.
10. R. Klatte, U. W. Kulisch, A. Wiethoff, Ch. Lawo, and M. Rauch. *C-XSC. A C++ Class Library for Extended Scientific Computing*. Springer-Verlag, Berlin, 1993.
11. Walter Krämer and Werner Hofschuster. C-XSC 2.0: A C++ library for extended scientific computing. In René Alt, Andreas Frommer, R.B̃aker Kearfott, and Wolfram Luther, editors, *Numerical Software With Result Verification. International Dagstuhl Seminar, Dagstuhl Castle, Germany, January 19–24, 2003. Revised papers.*, volume 2991 of *Lecture Notes in Computer Science*, pages 15–35. Springer, Berlin, 2004.
12. Walter Krämer and Evgenija D. Popova. Zur Berechnung von verlässlichen Außen- und Inneneinschließungen bei parameterabhängigen linearen Gleichungssystemen. *Proc. Appl. Math. Mech.*, 4:670–671, 2004.
13. R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4:187–201, 1969.
14. R. E. Moore. A test for existence of solutions to nonlinear systems. *SIAM J. Numer. Anal.*, 14:611–615, 1977.
15. Evgenija D. Popova. Parametric interval linear solver. *Numerical Algorithms*, 37:345–356, 2004.
16. S. M. Rump. Solving algebraic problems with high accuracy. In W.L Miranker and E. Kaucher, editors, *A New Approach to Scientific Computation*, volume 7 of *Comput. Sci. Appl. Math.*, pages 51–120, New York, 1983. Academic Press.
17. S. M. Rump. Verification methods for dense and sparse systems of equations. In J. Herzberger, editor, *Topics in Validated Computations*, volume 5 of *Stud. Comput. Math.*, pages 63–135, Amsterdam, 1994. Elsevier.
18. S. M. Rump. Expansion and estimation of the range of nonlinear functions. *Math. Comput.*, 65(216):1503–1512, 1996.
19. S. M. Rump. INTLAB – INTerval LABoratory. In T. Csendes, editor, *Developments in Reliable Computing*, pages 77–104, Dordrecht, 1999. Kluwer Academic Publishers.
20. P. D. Seymour and T. Zaslavsky. Averaging sets: A generalization of mean values and spherical designs. *Adv. Math.*, 52:213–240, 1984.
21. Ian H. Sloan and Robert S. Womersley. Extremal systems of points and numerical integration on the sphere. *Advances Comp. Math.*, 21:102–125, 2004.
22. Ian H. Sloan and Robert S. Womersley. A variational characterization of spherical designs. *J. Approx. Theory*, 159:308–318, 2009.
23. Steve Smale. Mathematical problems for the next century. *Mathematical Intelligencer*, 20:7–15, 1998.
24. Robert S. Womersley. Interpolation and cubature on the sphere. `web.maths.unsw.edu.au/~rsw/Sphere/Extremal/New/index.html`.