In search for good Chebyshev lattices

Koen Poppe *Ronald Cools*

Report TW 598, July 2011

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

In search for good Chebyshev lattices

Koen Poppe *Ronald Cools*

Report TW 598, July 2011

Department of Computer Science, K.U.Leuven

Abstract

Recently we introduced a new framework to describe some point sets used for multivariate integration and approximation (Poppe Cools, BIT Numerical Mathematics, 2011), which we called Chebyshev lattices. The associated integration rules are equal weight rules, with corrections for the points on the boundary. In this text we detail the development of exhaustive search algorithms for good Chebyshev lattices where the cost of the rules, i.e., the number of points needed for a certain degree of exactness, is used as criterium. Almost loopless algorithms are considered to avoid dependencies on the rank of the Chebyshev lattice and the dimension. Also, several optimisations are applied: reduce the vast search space by exploiting symmetries, lower the cost of the point set creation and minimise the cost of the degree verification. The concluding summary of the search results indicates that higher rank rules in general are better and that the blending formulae due to Godzina lead to the best rules within the class of Chebyshev lattices: no better rules have been found in the searches conducted in up to 5 dimensions.

Keywords : Chebyshev lattice rules, Morrow-Patterson points, Padua points, Godzina blending, computational search, symmetrie reduced search space, loopless algorithms.
MSC : Primary : 65D32, Secondary : 68R05, 68P10.

In search for good Chebyshev lattices

Koen Poppe and Ronald Cools

Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A,
B-3001 Heverlee, Belgium,

Summary. Recently we introduced a new framework to describe some point sets used for multivariate integration and approximation [3], which we called Chebyshev lattices. The associated integration rules are equal weight rules, with corrections for the points on the boundary. In this text we detail the development of exhaustive search algorithms for *good* Chebyshev lattices where the cost of the rules, i.e., the number of points needed for a certain degree of exactness, is used as criterium. Almost loopless algorithms are considered to avoid dependencies on the rank of the Chebyshev lattice and the dimension. Also, several optimisations are applied: reduce the vast search space by exploiting symmetries, lower the cost of the point set creation and minimise the cost of the degree verification. The concluding summary of the search results indicates that higher rank rules in general are better and that the blending formulae due to Godzina lead to the best rules within the class of Chebyshev lattice rules: no better rules have been found in the searches conducted in up to 5 dimensions.

1 Introduction

1.1 Motivation

Recently Clenshaw-Curtis integration was revived [14, 15]. This technique for 1-dimensional integration is based on a cosine mapped trapezoidal rule. The question that motivated this research is: can a similar transform applied to a lattice rule be beneficial for multivariate integration and approximation. In [3] we presented the Chebyshev lattice rules as generalising framework for cubature rules suited for integration with Chebyshev weight function. This setting arises naturally when determining the coefficients of a Chebyshev approximation of a multivariate function. Because of the limited number of parameters, the Chebyshev lattice notation enables an exhaustive search in moderate dimensions. We are interested in *good* cubature rules in the sense that they require a small number of points, and thus function evaluations, to give exact results for the weighted integral of polynomials up to a certain degree. We hope that, as known (near-) optimal point sets fit into the framework, good cubature rules in moderate dimensions may also be found.

1.2 Classical lattices

Lattice rules are a well known family of quasi-Monte Carlo methods for the approximation of s-dimensional integrals (see, e.g., [13, 2] and their references). They are based on point sets that can easily be described as a linear combination of $k \leq s$ generating vectors. More specifically they are based on so-called integration lattices: a discrete subset of the real space $\mathbb{R}^s$ that is closed under addition and subtraction that contains the integer points as a subset. Hence the point set can be described as

$$\left\{ \sum_{j=1}^{k} \frac{\ell_j \mathbf{z}_j}{d_j} \colon \ell_j \in \mathbb{Z}, d_j \in \mathbb{N}_0 \text{ and } \mathbf{z}_j \in \mathbb{Z}^s \text{ for } j = 1, \ldots, k \right\}, \tag{1}$$

where we use curly braces only to denote the *set* of points, $\mathbb{Z}$ is the set of integers and $\mathbb{N}_0 := \{1, 2, \ldots\}$. The associated lattice rule uses a set of points $\Lambda = \{\mathbf{x}\} \subset [0,1)^s$, where the components of $\mathbf{x}$ consist of the fractional part of the lattice points, to approximates the integral of $f(\mathbf{x})$ on the domain $[0,1]^s$

$$Q[f] := \frac{1}{N} \sum_{\mathbf{x} \in \Lambda} f(\mathbf{x}) \quad \approx \quad \int_{[0,1]^s} f(\mathbf{x})\, \mathsf{d}\mathbf{x} =: I[f]. \tag{2}$$

Note that N, the number of points in (2), follows from the generating vectors, more specific $N = |\det \mathbf{G}|^{-1}$ where $\mathbf{G}$ is a rational generator matrix of the lattice (1) [7].

There are many quality criteria for lattice rules in use, see [2] for a survey. One of these is the *total trigonometric degree*. This is defined as the maximal $n = |\mathbf{h}| := \sum_{r=1}^{s} |h_r|$ for which all trigonometric functions $f(\mathbf{x}) = \prod_{r=1}^{s} \exp(2\pi\imath\, h_r x_r)$ are integrated exactly, i.e., (2) becomes an equality. This allows for a ranking of rules: *good lattice rules* have a small number of points N compared to other rules with the same trigonometric degree and thus require fewer function evaluations.

1.3 Chebyshev lattices

Chebyshev lattices of rank k, as introduced in [3], are based on a cosine mapping of a classical lattice with the same k generating vectors $\mathbf{z}_j \in \mathbb{Z}^s$ and denominators $d_j \in \mathbb{N}_0$, a fixed offset vector $\mathbf{z}_\Delta \in \mathbb{Z}^s$ and denominator $d_\Delta \in \mathbb{N}_0$:

$$\chi := \left\{ \cos\left(\pi \left(\sum_{j=1}^{k} \frac{\ell_j \mathbf{z}_j}{d_j} + \frac{\mathbf{z}_\Delta}{d_\Delta} \right) \right) : \ell_j \in \mathbb{Z} \text{ for } j = 1, \ldots, k \right\}. \tag{3}$$

The point sets (3) were developed in the context of multivariate integration on hypercubes $C_s := [-1,1]^s$ using a Chebyshev approximation of the integrand. Evaluating the coefficients of the approximation leads to integrals with Chebyshev weight function $\omega(\mathbf{x}) := \pi^{-s} \prod_{r=1}^{s} (1 - x_r^2)^{-\frac{1}{2}}$ that, due to hyperinterpolation theory [12], can be replaced with a suitable cubature rule in which we use the points from (3):

$$Q[f] := \sum_{\mathbf{x} \in \chi} w_{\mathbf{x}} f(\mathbf{x}) \quad \approx \quad \int_{C_s} f(\mathbf{x})\, \omega(\mathbf{x})\, \mathsf{d}\mathbf{x} =: I[f]. \tag{4}$$

To avoid a periodicity requirement, and inherently through the cosine mapping, χ possibly includes points on all boundaries. The cubature rule (4) is therefore an equal weight rule, with corrections for the points on the boundary. The weights are still known explicitly, i.e., it is not necessary to solve a system of equations to obtain their values. To correct for points on the boundary, a scaling of the weights is needed: in three dimensions, points on faces, edges and vertices will have weights proportional to $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{8}$. Using the conditional function $\phi(\text{condition})$, which evaluates to 1 if the 'condition' is true, 0 otherwise, the weight $w_{\mathbf{x}}$ can be written as

$$w_{\mathbf{x}} := \frac{\tilde{w}_{\mathbf{x}}}{\tilde{W}}, \quad \text{where} \quad \tilde{w}_{\mathbf{x}} := 2^{-\sum_{r=1}^{s} \phi(|x_r|=1)} \quad \text{and} \quad \tilde{W} := \sum_{\mathbf{x} \in \chi} \tilde{w}_{\mathbf{x}}, \tag{5}$$

in which the normalisation factor $\tilde{W}$ ensures exactness of (4) for a constant function.

As with classical lattices, the quality of a cubature rule (4) can be expressed in terms of its degree, but in this weighted setting we use the *total algebraic degree*. If (4) is an equality for all polynomial functions $f(\mathbf{x}) = \prod_{r=1}^{s} x_r^{h_r}$, where $|\mathbf{h}| \leq n$, the cubature rule is said to have a degree n. Similar to classical lattice rules, *better* Chebyshev lattice rules require less points to attain a certain degree n.

It is important to note that, in contrast to the classical lattice rules, there is no closed relation between the parameters of the Chebyshev lattice and the number of points N. Also, due to the folding of the cosine mapping, it is not guaranteed that *good* classical lattice parameters lead to *good* Chebyshev lattice rules and vice versa.

We showed [3] that most cubature point sets for the integrals (4) with Chebyshev weight function can be written as a Chebyshev lattice rule. Godzina's *blending point set* [5], for example, an explicit s-dimensional point set with given degree n, leads to the following full rank ($k = s$) Chebyshev lattice rule for $n = 4\nu - 1$ ($\nu \in \mathbb{N}_0$):

$$\begin{array}{l} \mathbf{z}_1 = [1,1,1 \cdots 1,1] \\ \mathbf{z}_2 = [0,2,0 \cdots 0,0] \\ \quad \vdots \qquad\qquad \ddots \\ \mathbf{z}_s = [0,0,0 \cdots 0,2] \end{array} \quad \text{with} \quad \begin{cases} d_1, \ldots, d_s = d_\Delta = 2\nu, \\ \mathbf{z}_\Delta = [0,1,0,1,\cdots]. \end{cases}$$

1.4 Computer search

It is obvious that Chebyshev lattice rules are described by a fixed number of parameters. This search space depends on the rank k and the dimension s and is bounded: due to the periodicity of the cosine mapping, the components of the generating vectors can be reduced modulo their denominators. Without loss of generality, we consider all denominators equal, i.e., $d_1 = \ldots = d_k = d$ and this common denominator d can be seen as the third search parameter. Note that any given Chebyshev lattice can be reformulated with equal denominators by taking d equal to the least common multiple of all denominators $d_1, \ldots, d_k$. This simplifies bookkeeping but postpones specific rules to higher values of d and thus more expensive searches. For example, a rank-2 Chebyshev lattice with $d_1 = 5$ and $d_2 = 7$ will only be found when $d = 35$.

Each combination of parameters s, k and d leads to a different search space. Our aim is to find the best Chebyshev lattices, i.e., the rules that require the lowest number of points to attain a certain degree of accuracy. Therefore, the programs keep track of the best Chebyshev lattice parameters, i.e., the generating vectors, denominator and the number of points for each degree and replaces them only if a rule with less points is found. These search process concepts are illustrated in Listing 1.

It is obvious that the search from Listing 1 can be done in parallel for the number of dimensions s, the rank k and the denominator d. This will require some post-processing, to combine all the rules and find the 'globally' best ones, but can be solved easily by storing the rules into a small database and by performing simple queries to extract the best Chebyshev lattice rules.

Whenever possible, the search programs have been made invariant of the three search parameters. To illustrate this, consider the most rudimentary way to iterate over rank-1 generators in s dimensions using s nested loops, one for each component of the generating vector. This is straightforward but limits the applicability of the program to a fixed s. To avoid this dependency, we have been exploring *loopless* and *almost loopless* algorithms [4, 6] that produce the same result with only one or two nested loops, independent of the number of dimensions. This way, one could re-write the rank-1 search so that the number of dimensions is just an input parameter: no code must be changed to run the search for another number of dimensions. This is clearly less error prone than explicit loops and could arguably be even slightly faster.

In the following three sections, we provide more details on the actual search programs and optimisations that have been used in the highlighted steps from Listing 1. After that, actual search results in up to 5 dimensions are compared to known point sets. Section 6 concludes this paper.

Listing 1. Conceptual overview of the search for good Chebyshev lattice rules. Sections 2 to 4 elaborate on the lines that are indicated in the algorithmic overview.

```
for s = 2,3,...
  catalog = [ ]
  for k = 1,...,s
    for d = 1,2,... until time budget for this s exhausted
      for each {z_1,...,z_k} in the search space                    ⇝ Section 2
          create the point set; let N be the number of points       ⇝ Section 3
          determine the algebraic total degree n                    ⇝ Section 4
        if( is_empty(catalog[n]) || N < catalog[n].N ) then
          catalog[n] = {z_1,...,z_k,d,N}
        end if
      end for
    end for
  end for
  report the rules from catalog
end for
```

2 Reducing the search space

2.1 Exploiting symmetry

The search space for generating vectors of s-dimensional, rank-k Chebyshev lattices where the components of each j-th generating vector component belong to $\{0,1,\ldots,d\}$, has $(d+1)^{ks}$ elements. However, symmetries can be exploited to reduce this huge search space. Due to the symmetry of C_s and ω, the components of a point set can be permuted without influencing the degree. The same reasoning allows us to reorder the components of the generating vectors, which of course is not limited to rank-1. The higher rank case, i.e., $k>1$, can also exploit the invariance of the Chebyshev lattice with respect to the ordering of the k generating vectors.

Let us first focus on s-dimensional rank-1 Chebyshev lattices: consider the integer search space where all generating vectors $\mathbf{z}_1$ live in. The number of symmetries can be related to the number of distinct component values of this vector, here denoted by t. The most obvious example are the $d+1$ vectors that have only one independent component, thus $t=1$. This case can be summarised with a parametric description $\mathbf{z}_1=(\mathtt{A},\ldots,\mathtt{A})$. For $t>1$, several parameter descriptions can be made, e.g., for $s=3$ and $t=2$ there is $\mathbf{z}_1=(\mathtt{A},\mathtt{A},\mathtt{B})$ and $\mathbf{z}_1=(\mathtt{A},\mathtt{B},\mathtt{B})$ in which the parameters $\mathtt{A}<\mathtt{B}$ It should be clear that if all vectors satisfying these descriptions are checked, it is no longer necessary to verify $(\mathtt{A},\mathtt{B},\mathtt{A})$, $(\mathtt{B},\mathtt{A},\mathtt{A})$, $(\mathtt{B},\mathtt{B},\mathtt{A})$ or $(\mathtt{B},\mathtt{A},\mathtt{B})$ because of the cubature rules degree's invariance with respect to coordinate permutations. This leads to an asymptotical reduction of $\frac{1}{s!}$ for $d\to\infty$, but note that the reduced space, with $\frac{1}{s!}(d+1)^{ks}$ elements, still explodes for growing s, k or d.

We define the *unique* generating vectors as the sets of generating vectors that are in the reduced search space in which all of the above invariances have been excluded. To see how many of those generating vectors there are, we first observe that the entire search space in s dimensions can be decomposed as

$$(d+1)^s=\sum_{t=1}^{s}\left(\underbrace{\binom{d+1}{t}}_{\mathrm{I}}\cdot\underbrace{\sum_{\mathbf{g}}\frac{s!}{\prod_{v=1}^{t}g_v!}}_{\mathrm{II}}\right). \tag{6}$$

In this, the I-part represents the number of generators one can construct with t-different parameters in $\{0,\ldots,d\}$ and the II-part denotes the number of symmetries that should also be included to get the entire space (this is also the number of permutations of the multi-set with occurrence counts $\mathbf{g}$). Examples of the vectors $\mathbf{g}$ are given in Table 1 and a proof of (6) for $s\leq 4$ is given in the Appendix.

Using this formalism, the size of the search space for rank-1 rules can be written explicitly by taking only one symmetry setting per parameter description. This replaces the II-part in (6) by $\#\mathbf{g}$, the number of $\mathbf{g}$ vectors, and leads to (see Appendix)

$$U_{s,d,k=1}=\sum_{t=1}^{s}\left(\binom{d+1}{t}\cdot\#\mathbf{g}\right)=\sum_{t=1}^{s}\left(\binom{d+1}{t}\cdot\binom{s-1}{s-t}\right). \tag{7}$$

For higher ranks, a similar approach can be followed, but note that coordinate changes need to be considered for all k generating vectors together. A similar parametric approach as above can be used to describe the vectors. In two dimensions, rank $k=2$, this leads to the following sets of vectors where $\mathtt{A}<\mathtt{B}<\mathtt{C}<\mathtt{D}$:

$$\mathbf{Z}=\left[\frac{\mathbf{z}_1}{\mathbf{z}_2}\right]=\left[\frac{\mathtt{A},\mathtt{A}}{\mathtt{A},\mathtt{B}}\right],\left[\frac{\mathtt{A},\mathtt{B}}{\mathtt{A},\mathtt{C}}\right],\left[\frac{\mathtt{A},\mathtt{B}}{\mathtt{B},\mathtt{A}}\right],\left[\frac{\mathtt{A},\mathtt{B}}{\mathtt{C},\mathtt{A}}\right],\ldots,\left[\frac{\mathtt{A},\mathtt{B}}{\mathtt{C},\mathtt{D}}\right]. \tag{8}$$

The second symmetry for higher ranks is the order of the k vectors. Obviously, permuting them does not change the Chebyshev lattice. Also, because the search focusses on a specific rank, sets of vectors with a linear dependency can be discarded.

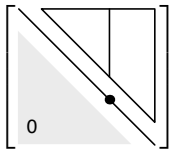
The parametric representation of the generators provides us with a flexible and memory friendly way of avoiding symmetries without explicitly storing all visited generators. Creating the parametric descriptions requires only a moderate $\mathcal{O}\big((ks)^{ks}\big)$ memory locations. It is important to see that this number is independent of d: the descriptions can thus be calculated once and re-used for every value of the denominator (skipping descriptions where $t>k(d+1)$).

Our current programs use an almost loopless algorithm to generate the prescriptions for arbitrary rank k and dimension s and stores them on disk. The actual search then reads these descriptions one by one and uses a second almost loopless algorithm to generate the values for the parameters $\mathtt{A}, \mathtt{B}, \ldots$ Note that the loopless property dramatically simplifies this process, because the number of parameters t is now just an input argument of the algorithm: t may vary between descriptions without adding any complexity to the program.

2.2 Hermite normal form

Following the discussions in [13, Chapter 9], another way of avoiding spurious elements in the search space is by using the Hermite normal form. It is shown that integration lattices yield a full-rank matrix $\mathbf{H}$ which is derived from a generator matrix $\mathbf{Z} \in \mathbb{Z}^{s \times s}$ with on its rows the k generating vectors, extended with $s-k$ suitable unit vectors, using a unimodular transformation $\mathbf{U}$ so that $\mathbf{Z} = \mathbf{U}\mathbf{H}$. These Hermite matrices have a specific structure which can be exploited during the search process:

$$\mathbf{H} = [H_{i,j}] = \left[\begin{array}{c} \\ \\ \\ \\ \end{array}\right.$$



$$\text{where} \begin{cases} H_{i,j} = 0, & j < i, \\ 0 < H_{i,j} \leq d, & j = i, \\ 0 \leq H_{i,j} < H_{j,j}, & j > i. \end{cases} \tag{9}$$

To enumerate all $\mathbf{H}$ matrices, only $\frac{s(s+1)}{2}$ parameters are needed instead of s^2 when considering all generator matrices $\mathbf{Z}$. Using the specific structure of $\mathbf{H}$, the number of elements in this search space is thus only

$$U_{s,d} = \prod_{r=1}^{s} \sum_{\delta=0}^{d} \delta^{r-1}. \tag{10}$$

We have implemented a routine that iterates over all Hermite normal form matrices $\mathbf{H}$, given s and a denominator d. For this search, we loosened the fixed rank requirement and checked all sets of generating vectors *up to* a certain rank-k because this determines the number of parameters in $\mathbf{H}$.

(a) $s = 1$

t	Descr.	$\mathbf{g}$	$\frac{s!}{\prod_{v=1}^{t} g_v!}$	$\sum_{\mathbf{g}} \frac{s!}{\prod_{v=1}^{t} g_v!}$
1	A	[1]	1	1

(b) $s = 2$

t	Descr.	$\mathbf{g}$	$\frac{s!}{\prod_{v=1}^{t} g_v!}$	$\sum_{\mathbf{g}} \frac{s!}{\prod_{v=1}^{t} g_v!}$
1	A,A	[2]	1	1
2	A,B	[1,1]	2	2

(c) $s = 3$

t	Descr.	$\mathbf{g}$	$\frac{s!}{\prod_{v=1}^{t} g_v!}$	$\sum_{\mathbf{g}} \frac{s!}{\prod_{v=1}^{t} g_v!}$
1	A,A,A	[3]	1	1
2	A,A,B	[2,1]	3	6
	A,B,B	[1,2]	3	
3	A,B,C	[1,1,1]	6	6

(d) $s = 4$

t	Descr.	$\mathbf{g}$	$\frac{s!}{\prod_{v=1}^{t} g_v!}$	$\sum_{\mathbf{g}} \frac{s!}{\prod_{v=1}^{t} g_v!}$
1	A,A,A,A	[4]	1	1
2	A,A,A,B	[3,1]	4	14
	A,A,B,B	[2,2]	6	
	A,B,B,B	[1,3]	4	
3	A,A,B,C	[2,1,1]	12	36
	A,B,B,C	[1,2,1]	12	
	A,B,C,C	[1,1,2]	12	
4	A,B,C,D	[1,1,1,1]	24	24

Table 1. Examples of the descriptions of a s-dimensional grid with components in $\{0, \ldots, d\}$ with their occurrence vector $\mathbf{g}$. The last column lists the specific values of part II in (6).

3 Reducing the point set creation cost

When creating a Chebyshev lattice point set, duplicates must be avoided. This is somehow expensive because for a rank-k Chebyshev lattice with common denominator d, $\bar{N} = (d+1)^k$ s-dimensional points must be verified to be distinct. Using a balanced tree structure, a red-black tree in our programs, the number of scalar comparisons in this operation can be reduced from $\mathscr{O}\left(s\bar{N}^2\right)$ to $\mathscr{O}(s\bar{N}\log\bar{N})$, but further improvements are possible. With the common denominator d, it is easy to see that points from the Chebyshev lattice are derived from an integer vector $\mathbf{y}_\ell$. Rewriting the points from (3) as

$$\mathbf{x}_\ell = \cos\left(\frac{\pi}{d}\mathbf{y}_\ell\right) \qquad \text{with} \qquad \mathbf{y}_\ell := \sum_{j=1}^{k} \ell_j \mathbf{z}_j + \mathbf{z}_\Delta \tag{11}$$

clearly shows this. Moreover, due to the periodicity of the cosine, this integer vector $\mathbf{y}_\ell$ can be reduced, component by component, so that it still produces the same point $\mathbf{x}_\ell$. Using $\tilde{\mathbf{y}}_\ell$ for this reduced integer vector and 'minmod$_d$' for the element-wise reduction so that $\tilde{\mathbf{y}}_\ell \in [0,d]^s$, we can write the points as

$$\mathbf{x}_\ell = \cos\left(\frac{\pi}{d}\mathbf{y}_\ell\right) =: \cos\left(\frac{\pi}{d}\tilde{\mathbf{y}}_\ell\right) \quad \text{with} \quad \tilde{\mathbf{y}}_\ell := \text{minmod}_d\left(\mathbf{y}_\ell\right). \tag{12}$$

It suffices to compare s-dimensional integer vectors $\tilde{\mathbf{y}}_\ell$, instead of floating point vectors, to see whether a given point is already in the set. And, since d is rather small (less than a few hundred), it might be possible to compress the vectors even more. One way is to use integers with smaller ranges to represent $\tilde{\mathbf{y}}_\ell$ but combining the component values using a multi-radix notation proved to be even quicker. With sufficiently large integers, $\tilde{\mathbf{y}}_\ell$ can be stored as a scalar

$$\gamma_\ell = \sum_{r=1}^{s} \tilde{\mathbf{y}}_{\ell,r}(d+1)^r, \tag{13}$$

which is significantly faster in comparisons than the loop that is needed to go over all the components. Note that the tree also requires less memory as it only has to store γ_ℓ values to determine whether a point is a duplicate or not.

4 Reducing the degree verification cost

4.1 Evaluation of the basis function

In order to check the degree of a given point set and weights, the cubature rule (4) is evaluated for polynomials with increasing degree. It seems that, without prior knowledge about the point set, not much can be done to accelerate this step: all polynomials for increasing degrees must be verified. However, using Chebyshev polynomials $T_{\mathbf{h}}(\mathbf{x}) := \prod_{r=1}^{s}\cos\left(h_r \arccos(x_r)\right)$ simplifies the evaluation: it is obvious that $\arccos(x_r)$ can be precomputed, but, because all points will originate from a Chebyshev lattice, using $\tilde{\mathbf{y}}_\ell$ from (12) leads to

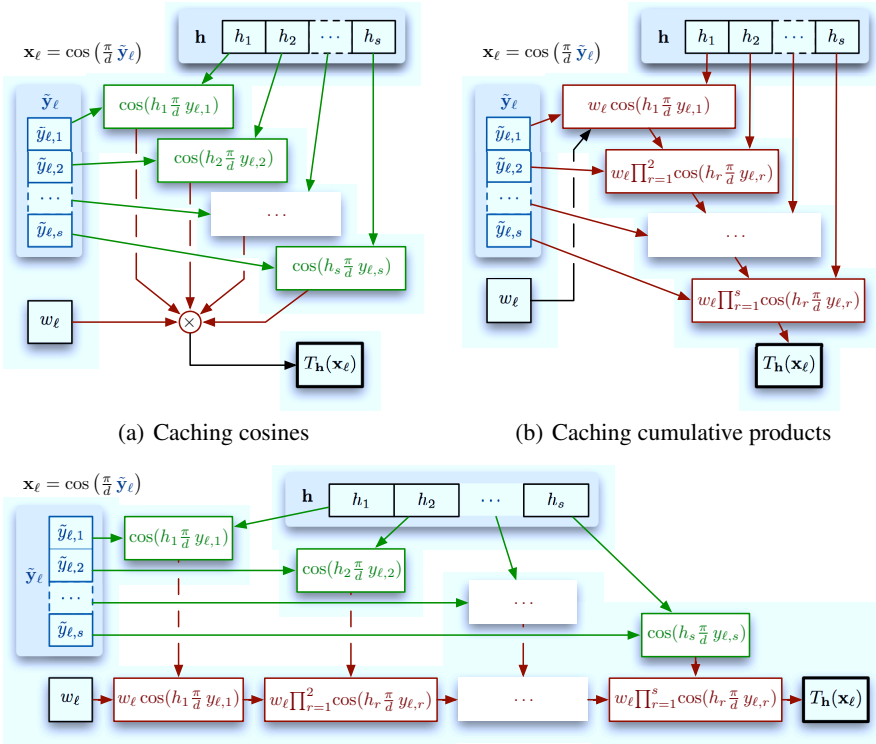
$$T_{\mathbf{h}}(\mathbf{x}_\ell) = \prod_{r=1}^{s}\cos\left(h_r \arccos\left(\cos\left(\frac{\pi}{d}y_{\ell,r}\right)\right)\right) = \prod_{r=1}^{s}\cos\left(h_r\frac{\pi}{d}\tilde{y}_{\ell,r}\right) \tag{14}$$

and so the evaluation of the arccosine of the cosine can be avoided.

For each degree n, all polynomials $T_{\mathbf{h}}$ with $|\mathbf{h}| = n$ must be verified, but *the order* in which this is done does not matter. If explicit loops are used, $\cos(h_r\frac{\pi}{d}\tilde{y}_{\ell,r})$ will only change when h_r changes. By storing these product terms for each h_r (see Figure 1(a)), the computational complexity, in terms of the number of cosine evaluations, is $\mathscr{O}\left(\sum_{r=1}^{s}\binom{n}{r}N\right)$ for the evaluation for N points, instead of the $\mathscr{O}\left(s\binom{n}{s}N\right)$ a naive implementation would require.

Alternatively, the number of floating point operations can be reduced by storing cumulative products of the cosines, as illustrated in Figure 1(b). For example, if we know that only h_s changes, the repeated calculations of the cumulative product $\prod_{r=1}^{s-1}\cos(h_r\frac{\pi}{d}\tilde{y}_{\ell,r})$ from (14) can be cached easily.

These two ideas can also be combined, leading to the diagram in Figure 1(c). However, experiments for moderate dimensions ($s \leq 10$) have shown that this third variant does not provide an additional decrease in execution time. Caching the cumulative product from Figure 1(b) provides the fastest results and can be seen to be more cache friendly because of memory locality.



(a) Caching cosines

(b) Caching cumulative products

(c) Caching both cosines and cumulative products

Fig. 1. Three ways of reducing the evaluation complexity of $T_{\mathbf{h}}(\mathbf{x}_\ell)$ in (14). For the order in which the $\mathbf{h}$'s are generated, variant (b) is the fastest, followed by (a). Variant (c) is slightly slower than (b) but requires more bookkeeping. For clarity, the diagrams show the evaluation in only one point. Evidently, this is vectorised over all points to benefit from pipelining.

4.2 Generation of the coefficients

This complexity can be reduced even further using an algorithm that generates the coefficients $\mathbf{h}$ in a way that minimises component changes because this increases the efficiency of the caches mentioned before. Of course, explicit loops already do this but as explained before we have been exploring almost loopless algorithms to avoid the dependency on the number of dimensions.

Our almost loopless algorithm, based on [4], generates all $\mathbf{h}$'s for a certain degree n using only two component changes per $\mathbf{h}$. This is optimal: less changes would violate the fixed degree assumption.

4.3 Special note on $n = 1$

As exactness for a constant ($n = 0$) is guaranteed by the scaling of the weights, the first degree that must be checked is $n = 1$. Research has shown that a significant part of the generators fail for this degree. Using the fact that all s coefficients $\mathbf{h}$ with $|\mathbf{h}| = 1$ have only one non-zero element, this degree can be verified using only sN cosines in total (N is the number of points in the Chebyshev lattice) compared to s^2N cosines if these zero elements are not exploited. Unrolling for $n = 1$ speeds up some searches with up to a factor 3.

4.4 Incorporating knowledge of the point set

Orthogonal to the above, another way to improve the complexity of the degree verification is the use of symmetry information. If it is known in advance that a point set equals its reflection around any $x_r = 0$, only even degrees must be checked in that direction (odd degrees are guaranteed). This proves very effective in reducing the number of

polynomials in the verification, but for now symmetry information is not available a such. As the generating vectors are known, they do provide information about the symmetry. However, due to the folding-like operations of the element-wise cosine transform in the Chebyshev lattices, it is unclear how to derive this kind of symmetry properties directly.

5 Search results

All results from the different search programs are summarised in Table 2. This shows that the best point sets do not improve the results of Godzina given at the end of Section 1.3. Therefore we measure the cost for the rules, i.e., the number of points, relative to those of Godzina as can be seen in Figure 2. In up to four dimensions we have found point sets equivalent to those due to Godzina, those cases are indicated with a circle on the graphs. Note that although Godzina's point set is generally described with a full rank Chebyshev lattice (i.e., $k = s$), for a degree $n = 1$, the denominator $d = 2$ and thus the rank of the rule drops to 1.

Although we searched for Chebyshev lattices without offset, i.e., $\mathbf{z}_\Delta = \mathbf{0}$, Figure 2 also includes results for the shifted Chebyshev lattices that were found by adding shifts to the previously known rules. Often, by adding a shift, the number of points could be reduced a little, while retaining the degree.

In two dimensions, the Morrow-Patterson points [9] are known to be *optimal*: they achieve the theoretical lower bound for the number of points due to Möller [8]. As shown in [3], these optimal set are a specialisation of Godzina's point set. The best two-dimensional rank-1 rules in correspond to the Padua points [1]. They are non-optimal and require more points to achieve the same degree as those by Morrow-Patterson.

For $s = 3$, the point set described by Noskov [10], again instances of Godzina's rules, are found to be best. It also appears that good rank-2 rules are rather hard to find: within the self imposed bounds on the search time, for corresponding denominators, both good rank-1 and good rank-3 rules were found. The rank-2 rules however, seem to require larger denominators and thus more time than allowed. From the incomplete results of Figure 2(b), a first general observation can be made: except for $n = 1$, higher rank rules require less points for the same degree. This actually encourages us to pursue high rank searches, although this requires iterating in a much larger space.

Results in 4 dimensions (see Figure 2(c)) and preliminary ones for $s = 5$ (only shown in Table 2) support the observation that higher ranks require less points. Investigation of the actual point sets also shows a favour for grid-like structures. This is rather unwanted, because it corresponds to quickly growing number of points when the degree and the number of dimensions increases. Note however that this growth is intrinsic for the definition of the degree we have chosen here, so it cannot be eliminated completely.

6 Conclusion

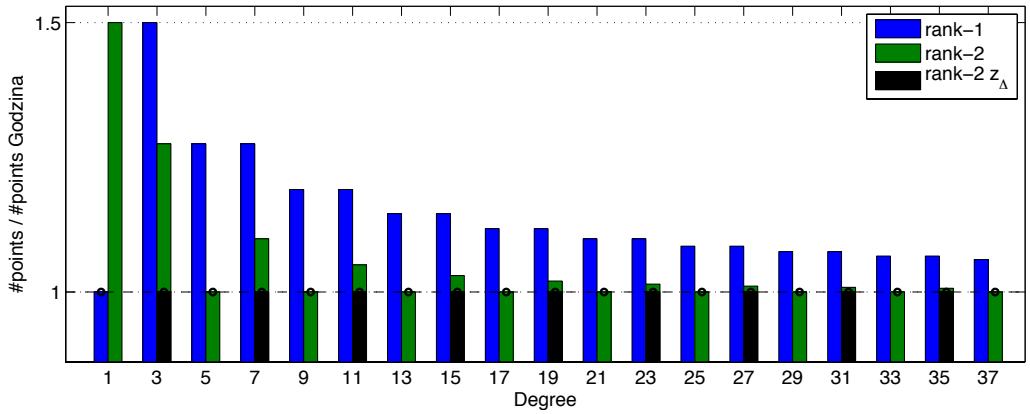
We have presented several approaches to search for good Chebyshev lattices and detailed the implementation and optimisation using, amongst other, (almost) loopless algorithms and caching structures. Our searches provide computational evidence that Godzina's point set is 'optimal' within all Chebyshev lattices: no better point sets were found so far. They are of full rank and require less points than lower rank results, which also appears to be a general observation in this context. An advantage of these not so commonly known blending point sets and rank-1 Chebyshev lattices is the ability to use the fast Fourier transform while creating Chebyshev approximations of a function. This might lead to efficient software for interpolation and integration in moderate dimensions [11].

With these results, we will not pursue the search for good Chebyshev lattices, although future work could include worst-case-error based criteria and possibly other definitions of the polynomial degree. Such quality criteria are required anyway if one wants to extend the applicability to higher dimensions.

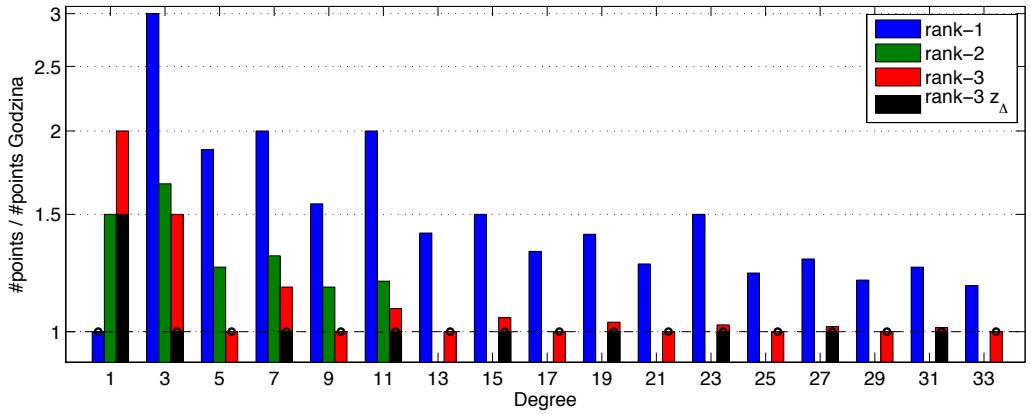
Acknowledgement

This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State Science Policy Office. The scientific responsibility rests with its authors.

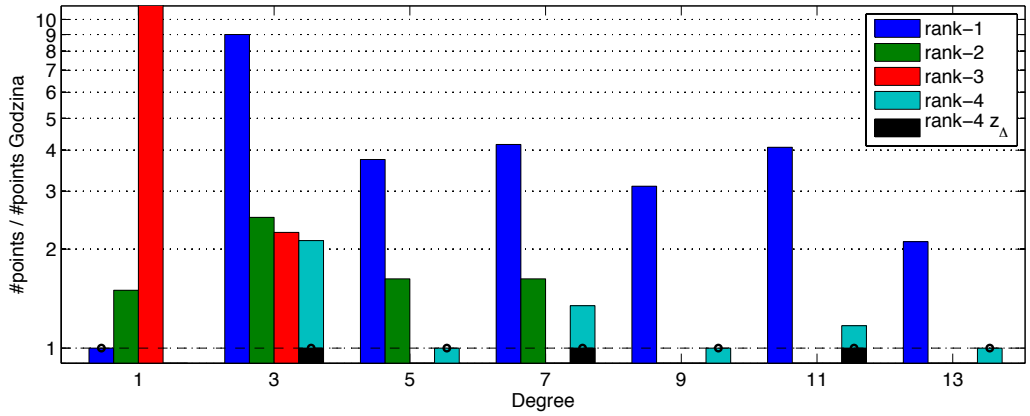
The authors acknowledge the support for this project by the Bijzonder Onderzoeksfonds of the Katholieke Universiteit Leuven.



(a) $s = 2$: for rank-2, the point sets due to Godzina's were found (some of them with an nonzero $\mathbf{z}_\Delta$), this corresponds to the well known Morrow-Patterson point set [9].



(b) $s = 3$: the best points sets are those described by Noskov [10], another specialisation of Godzina's set. No results for $k = 2$ and $n \geq 13$ were found within the search time bounds.



(c) $s = 4$: as with three dimensions, rank-k rules where $1 < k < s$ appear harder to find. Therefore, several degrees do not have a rank-2 and/or rank-3 rule. Godzina's set is also found here.

Fig. 2. Search results for up to 4 dimensions. As Godzina's point set describes the best Chebyshev lattices found so far, we show the relative number of points relative to the number in Godzina's set. The graphs show the number of points as function of the degree for different ranks. The black bars on top of the full-rank rules indicate results for non-zero offset vectors $\mathbf{z}_\Delta$. A circle indicates that the specific rule found, is identical to a point set due to Godzina.

Table 2. Generating vectors corresponding to the Chebyshev lattices from Figure 2 without offset vector ($\mathbf{z}_\Delta = \mathbf{0}$). Note that the results for $s = 2$ are excluded here, because they correspond to the Padua ($k = 1$) and Morrow-Patterson ($k = 2$) points described in [3]. The same applies for $k = s$ where the results correspond to Godzina's point set. In 5 dimensions, only rank-1 Chebyshev lattices and one rank-2 have been found so far.

(a) $s = 3, k = 1$

n	$\mathbf{z}_1$			d	N
1	1	1	1	1	2
3	6	10	15	30	18
5	12	15	20	60	30
7	20	28	35	140	60
9	30	35	42	210	84
11	42	66	77	462	168
13	56	63	72	504	180
15	72	88	99	792	270
17	90	99	110	990	330
19	110	130	143	1430	462
21	132	143	156	1716	546
23	156	204	221	2652	819
25	182	195	210	2730	840
27	210	238	255	3570	1080
29	240	255	272	4080	1224
31	272	304	323	5168	1530
33	306	323	342	5814	1710

(b) $s = 3, k = 2$

n	$\mathbf{z}_1$			$\mathbf{z}_2$			d	N
1	0	0	2	2	2	1	2	3
3	0	4	6	3	6	3	6	10
5	3	4	4	0	0	8	12	20
7	4	5	5	0	0	10	20	39
9	6	25	6	30	25	6	30	63
11	7	18	7	21	18	7	42	100

(c) $s = 4, k = 1$

n	$\mathbf{z}_1$				d	N
1	1	1	1	1	1	2
3	30	42	70	105	210	72
5	60	84	105	140	420	120
7	140	180	252	315	1260	300
9	210	330	385	462	2310	504
11	462	546	858	1001	6006	1176
13	504	616	693	792	5544	1080

(d) $s = 4, k = 2$

n	$\mathbf{z}_1$				$\mathbf{z}_2$				d	N
1	0	0	0	2	2	2	2	1	2	3
3	0	6	4	0	3	3	6	2	6	20
5	3	4	4	9	3	12	4	3	12	52
7	4	5	5	12	4	15	5	4	20	117

(e) $s = 4, k = 3$

n	$\mathbf{z}_1$				$\mathbf{z}_2$				$\mathbf{z}_3$				d	N
1	1	0	0	1	0	1	0	0	0	0	1	0	2	27
3	3	3	3	2	0	6	0	4	0	0	6	4	6	18

(f) $s = 5, k = 1$

n	$\mathbf{z}_1$					d	N
1	1	1	1	1	1	1	2
3	210	330	462	770	1155	2310	432
5	420	660	924	1155	1540	4620	720
7	1260	1540	1980	2772	3465	13860	1800
9	2520	3080	3465	3960	5544	27720	3240
11	6006	7854	9282	14586	17017	102102	10584
13	5544	6552	8008	9009	10296	72072	7560

(g) $s = 5, k = 2$

n	$\mathbf{z}_1$					$\mathbf{z}_2$					d	N
1	0	0	0	0	2	2	2	2	2	1	2	3

Appendix: Proof of identity (6) and (7)

The sum of a general function $G(\mathbf{g})$ over all $\mathbf{g}$'s is equivalent to $t-1$ nested sums

$$\sum_{\mathbf{g}} G(\mathbf{g}) \equiv \sum_{g_1=1}^{s-(t-1)} \sum_{g_2=1}^{s-(t-2)-g_1} \sum_{g_3=1}^{s-(t-3)-g_1-g_2} \cdots \sum_{g_{t-1}=1}^{s-1-\sum_{\mu=1}^{t-2} g_\mu} G(\mathbf{g})\Big|_{g_t = s-\sum_{\mu=1}^{t-1} g_\mu}. \tag{15}$$

Therefore, part II from identity (6) can be rewritten into

$$\sum_{\mathbf{g}} \frac{s!}{\prod_{v=1}^{t} g_v!} = \sum_{g_1=1}^{s-(t-1)} \frac{1}{g_1!} \sum_{g_2=1}^{s-(t-2)-g_1} \frac{1}{g_2!} \sum_{g_3=1}^{s-(t-3)-g_1-g_2} \frac{1}{g_3!} \cdots \sum_{g_{t-1}=1}^{s-1-\sum_{\mu=1}^{t-2} g_\mu} \frac{s!}{g_{t-1}! \left(s - \sum_{\mu=1}^{t-1} g_\mu\right)!}.$$

We verified (6) computationally for $s \in [1,20]$ and $d \in [1,100]$. The proofs for $s \in [2,4]$ ($s = 1$ is trivial), as used in this paper, are straightforward. The bold numbers correspond to part II of (6) and are taken from Table 1. Hence we obtain:

$$
\begin{aligned}
\boxed{s=2}\quad & \tbinom{d+1}{1}\mathbf{1}+\tbinom{d+1}{2}\mathbf{2} \\
&= (d+1)+d(d+1) \\
&= d^2+2d+1=(d+1)^2,
\end{aligned}
$$

$$
\begin{aligned}
\boxed{s=3}\quad & \tbinom{d+1}{1}\mathbf{1}+\tbinom{d+1}{2}\mathbf{6}+\tbinom{d+1}{3}\mathbf{6} \\
&= (d+1)+3d(d+1)+(d-1)d(d+1) \\
&= d^3+3d^2+3d+1=(d+1)^3,
\end{aligned}
$$

$$
\begin{aligned}
\boxed{s=4}\quad & \tbinom{d+1}{1}\mathbf{1}+\tbinom{d+1}{2}\mathbf{14}+\tbinom{d+1}{3}\mathbf{36}+\tbinom{d+1}{4}\mathbf{24} \\
&= (d+1)+7d(d+1)+6(d-1)d(d+1)+(d-2)(d-1)d(d+1) \\
&= d^4+4d^3+6d^2+4d+1=(d+1)^4.
\end{aligned}
$$

Formula (7) can be proven easily for general s. Counting the $\mathbf{g}$'s corresponds to substituting $G(\mathbf{g}) \equiv 1$ in (15):

$$
\sum_{\mathbf{g}} 1 = \sum_{g_1=1}^{s-(t-1)} \sum_{g_2=1}^{s-(t-2)-g_1} \sum_{g_3=1}^{s-(t-3)-g_1-g_2} \cdots \sum_{g_{t-1}=1}^{-\sum_{\mu=1}^{s-1} g_\mu} 1 = \prod_{\nu=1}^{t-1} \frac{s-(t-\nu)}{\nu} = \binom{s-1}{t-1}.
$$

References

1. Caliari, M., De Marchi, S., Vianello, M.: Bivariate polynomial interpolation on the square at new nodal sets. Applied Mathematics and Computation **165**(2), 261–274 (2005)
2. Cools, R., Nuyens, D.: A Belgian view on lattice rules. In: A. Keller, S. Heinrich, H. Niederreiter (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2006, pp. 3–21. Springer (2008)
3. Cools, R., Poppe, K.: Chebyshev lattices, a unifying framework for cubature with the Chebyshev weight function. BIT Numerical Mathematics **51**, 275–288 (2011)
4. Ehrlich, G.: Loopless algorithms for generating permutations, combinations, and other combinatorial configurations. Journal of the ACM **20**(3), 500–513 (1973)
5. Godzina, G.: Dreidimensionale Kubaturformeln für zentralsymmetrische Integrale. Ph.D. thesis, Universität Erlangen-Nürnberg (1994)
6. Knuth, D.E.: Combinatorial Algorithms, *The Art of Computer Programming*, vol. 4 (2005-2009)
7. Lyness, J.N.: An introduction to lattice rules and their generator matrices. IMA Journal of Numerical Analysis **9**, 405–419 (1989)
8. Möller, H.: Lower bounds for the number of nodes in cubature formulae. Numerische Integration **45**, 221–230 (1979)
9. Morrow, C.R., Patterson, T.N.L.: Construction of algebraic cubature rules using polynomial ideal theory. SIAM Journal on Numerical Analysis **15**(5), 953–976 (1978)
10. Noskov, M.: Analogs of Morrow-Patterson type cubature formulas. Journal of Computation Mathematics and Mathematical Physics **30**, 1254–1257 (1991)
11. Poppe, K., Cools, R.: CHEBINT: a MATLAB/Octave toolbox for multivariate integration and interpolation based on Chebyshev approximations over a hypercube. In preparation
12. Sloan, I.H.: Polynomial interpolation and hyperinterpolation over general regions. Journal of Approximation Theory **83**(2), 238–254 (1995)
13. Sloan, I.H., Joe, S.: Lattice Methods for Multiple Integration. Oxford Science Publications (1994)
14. Trefethen, L.N.: Is Gauss quadrature better than Clenshaw-Curtis? SIAM Review **50**(1), 67–87 (2008)
15. Trefethen, L.N., Hale, N., Platte, R.B., Driscoll, T.A., Pachón, R.: Chebfun version 3. Oxford University (2009). http://www.maths.ox.ac.uk/chebfun/