UNIVERSITY OF CALIFORNIA, SAN DIEGO

Applications on Multi-Dimensional Sphere Packings: Derivative-Free Optimization

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Engineering Sciences with Specialization in Computational Science

by

Paul Belitz

Committee in charge:

Professor Thomas R Bewley, Chair
Professor Philip Gill
Professor Alison Marsden
Professor Daniel Tartakovsky
Professor Kraig Winters

2011

The dissertation of Paul Belitz is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Chair

University of California, San Diego

2011

EPIGRAPH

Practice, Practice. All is coming.

—Sri K. Pattabhi Jois

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

My father, Dietrich Belitz, for being the first in encouraging me to think.

My advisor, Prof. Thomas Bewley, for his amazing creativity, enthusiasm and unwavering support throughout my tenure at UCSD, and enthusiastic support of canine lab-mates.

Prof. Kristi Morgansen, for giving me that first opportunity. I would never be writing these words if it were not for her.

David Coleman for his friendship and support, despite my remarkably undignified entry into our friendship.

Robert Krohn for shared philosophy, interests, and taste, many hours of company, and that longboard.

Ralf Brunner, for a timely ride to the ER and extensive generosity.

Joe Cessna for setting and keeping the bar high, as well as consistently making lab meetings thoroughly entertaining.

My colleagues in the FCCR labs: Chris Colburn, David Zhang, Nick Morozovsky, Andrew Cavender, Chris Schmidt-Wetekam, Saam Ostovari.

Jenny Barrett, Atsuro Chiba, Steve Thompson, Michelle Hackett, Marne Greene, Kati Reszegi, Holly Gastil, Natasha Teran, and Tim Miller.

Monica Brown for more than can be written concisely.

PUBLICATIONS

Chapter 2 has been previously submitted for publication as Bewley, Belitz, &Cessna , "New Horizons in Sphere Packing Theory, Part I: Fundamental Concepts & Constructions, From Dense to Rare", *SIAM Review,* 2011, submitted.

Chapter 3 has been previously submitted for publication as Belitz & Bewley, "LABDOGS: LAttice-Based Derivative-Free Optimization via Global Surrogates", *Journal of Global Optimization,*2011, submitted

Chapter 4 is currently under preparation for submission for publication as Belitz & Bewley, "Λ-MADS: Mesh Adaptive Direct Search via Efficient Lattices", *SIAM Optimization,* 2011, submitted

VITA

2011	Ph. D. in Engineering Sciences with Specialization in Computational Science, University of California, San Diego
2008	M.S. in Mechanical and Aeronautical Engineering, University of California, San Diego
2006	B. S. in Aeronautics and Astronautics , University of Washington

PUBLICATIONS

Bewley, Belitz, &Cessna , "New Horizons in Sphere Packing Theory, Part I: Fundamental Concepts & Constructions, From Dense to Rare", *SIAM Review*, 2011, submitted.

Belitz & Bewley, "LABDOGS: LAttice-Based Derivative-Free Optimization via Global Surrogates", *Journal of Global Optimization*,2011, submitted

Belitz & Bewley, "Λ-MADS: Mesh Adaptive Direct Search via Efficient Lattices", *SIAM Optimization*, 2011, submitted

SELECT PROCEEDINGS

Belitz & Bewley, "'Checkers' - Highly Efficient Derivative-Free Optimization.", *Proceedings of the 38th AIAA Fluid Dynamics Conference and Exhibit*, AIAA-2008-4313, 2008.

Belitz & Bewley, "Efficient Derivative-Free Optimization.", *Proceedings of the 46th IEEE Conference on Decision and Control*, 5358-5363, 2007.

SELECT PRESENTATIONS

Invited Speaker and CoInstructor, Montestigliano Workshop 2011: Advanced Optimization Techniques in Fluid Mechanics, April 2011, Montestigliano, Italy

Paul Belitz, "Incorporating Regular Lattices in Derivative-Free Optimization", *General Electric, Albany NY*, 2010.

Paul Belitz, "Incorporating Regular Lattices and Accounting for Approximate Function Evaluations in Derivative-Free Optimization", *OPTIMA, University of Illinois, Urbana-Champaign*, 2009.

ABSTRACT OF THE DISSERTATION

Applications on Multi-Dimensional Sphere Packings: Derivative-Free Optimization

by

Paul Belitz

Doctor of Philosophy in Engineering Sciences with Specialization in Computational Science

University of California, San Diego, 2011

Professor Thomas R Bewley, Chair

The field of n-dimensional sphere packings is elegant and mature in its mathematic development and characterization. However, practical application of this powerful body of work is lacking. The line of research presented in this work explores the application of sphere packings to the field of derivative-free optimization. Chapter 2 reviews the essential results available in this field, then extends these results by: (a) assembling a catalog of key properties of the principle dense and rare sphere packings and nets available, including hundreds of values not previously known; (b) introducing and characterizing several new families of regular rare sphere packings and nets; and (c) developing a new algorithm for efficient solution of discrete Thompson problems, restricted to nearest-

neighbor points. These results are leveraged heavily in the applications addressed in Chapters 3 and 4. In particular, Chapter 3 builds from this presentation to develop a new algorithm for Lattice-Based Derivative-free Optimization via Global Surrogates (LABDOGS), leveraging dense sphere packings as an alternative to Cartesian grids to coordinate derivative-free searches. The LABDOGS algorithm provides a highly efficient, globally convergent optimization algorithm that requires nothing more than a definition of a feasible domain and a cost function handle. The LABDOGS algorithm demonstrates superior performance and convergence rates to its Cartesian-based competitors. Chapter 4 builds from the material of Chapter 2 and 3 to develop a highly efficient locally convergent derivative-free optimization algorithm called Λ-MADS, which builds from and improves upon the Mesh Adaptive Direct Search (MADS) class of optimization algorithms. The Λ-MADS algorithm offers an alternative to the Successive Polling substep of LABDOGS, providing a locally convergent pattern search algorithm that, unlike SP, offers good convergence behavior when challenging constraints on the feasible region are encountered. Λ-MADS inherits all the convergence characteristics of the best available MADS algorithms, while significantly improving convergence rates.

Chapter 1

Preface

The field of numerical optimization is utilized in a variety of applications in modern society. From mathematical and scientific research, to structural analysis and financial modeling, many fields have benefited from the widespread application of optimization theory. Generally speaking, numerical optimization is the process of locating the optimum of a cost function, which can be defined in virtually unlimited ways. Most often, the cost function is generated via experiment, simulation, model, or similar analysis. Some prototypical examples include the engineering of aircraft aerodynamics, designing automobile components, determining flight schedules, research of heart surgery simulation, jet engine development, gas laser tuning, and maximizing compiler performance. The impact of numerical optimization on modern technology, science, and engineering has been significant. In today's age of computing, the numerical tools for effective optimization have already had significant impact, and as computing resources continue to increase in power and further decrease in cost, the application of numerical optimization will have an increasing impact in a number of fields. It is a safe assumption that traditionally human-based decision making processes will further be phased out in favor of the computational power of modern computers.

Numerical optimization is naturally divided into two distinct classes of algorithms: those that utilize derivatives of the cost function to determine an appropriate direction of exploration, and derivative-free methods, which require nothing other than evaluations of the cost function at various points in parameter space.

Derivative-based optimization methods have a long history. The first calculus-

based analysis of function optima were performed by Pierre de Fermat and Joseph Luis Lagrange; iterative methods to numerically approach such optima were introduced by Issac Newton and Carl Friedrich Gauss. These methods are still the basis for many commonly used modern algorithms: Euler's method, appropriately formulated, gives the steepest descent method, often the first numerical algorithm presented in the field of optimization. Natural augmentations of Steepest Descent lead to the widespread benchmark Conjugate Gradient method, in which the negative gradient is augmented with previous derivative information to produce a more effective descent direction compared to Steepest Descent. Similarly, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is a modern adaptation of Newton's method, utilizing an approximated Hessian capturing the behavior of the cost function's higher derivatives to determine an effective step toward the function optimum at every step. The BFGS method is generally considered the most sophisticated algorithm for modern convex optimization algorithms, however, Conjugate Gradient methods and similar are often chosen for ease of implementation. Further work has generated modern derivative-based algorithm toolboxes, which include linear programming, quadratic programming, and nonlinear programming, all of which have found widespread use in industry and academia.

Methods not requiring the use of a function's derivatives have, perhaps surprisingly, a much shorter history. Not practically useful until the first computing devices were invented, modern derivative-free optimization algorithms were (arguably) born when George Danzig developed the Simplex algorithm in 1947, a method which generates a set of points in parameter space (a 'simplex'), surrounding the current best point, upon which the cost function is evaluated. The best point evaluated is located and a new simplex generated, allowing the algorithm to 'march' toward a local optimum. Similar derivative-free methods were soon developed, including those belonging to the Pattern Search class of algorithms of Hooke and Jeeves, identified in 1961. Like the original Simplex method, Pattern Search methods sample the cost function on a specific pattern of points surrounding the current best point, allowing the algorithm to converge to a local optimum of an appropriately behaved cost function. The geometric restrictions on patterns allowing algorithm convergence were identified by Yu (1979), who proved that on an appropriately smooth convex cost function, the use of a positive basis as the un-

derlying pattern is necessary to allow convergence. Further analysis has been performed by Torczon, Lagarias, Dolan, and others.

Outside the realm of the well-studied Pattern Search algorithms, less rigorous analysis, inspired by specific research challenges, resulted in several popular optimization schemes, including the classes of Genetic Algorithms and Simulated Annealing algorithms. The former was inspired by biology and is commonly used in computational biology, the latter inspired by metallurgy, and commonly utilized in modern engineering applications. Unlike traditional pattern search methods, both attempt to solve nonconvex problems via stochastic exploration steps. Due to the inherent difficulty of such problems, few other effective algorithms have been established, making SA and Genetic algorithms quite popular in certain fields.

Generally speaking, when applicable, derivative-based methods offer superior performance to the sometimes unsophisticated derivative-free methods. By locating an effective descent direction, modern derivative-based methods allow convergence to optima of extremely high-dimensional cost functions while not incurring a prohibitive computational expense. An excellent example is the application of Conjugate Gradient optimizations to optimal control problems, where the derivative-based algorithm allows optimization over a parameter space with a dimension greater than one million. By comparison, deriative-free methods, no matter how efficient or sophisticated, are more heavily penalized by the curse of dimensionality; that is, as the dimension of the cost function becomes greater, the difficulty of the optimization problem becomes great very quickly, relegating this class of methods to more moderate dimensions. Despite this intrinsic limitation, there are many cost functions of great importance which cannot be solved via derivative-based methods. These include, for example, any cost function that is defined by physical experiment, where it is impossible to calculate the derivative of the cost function. Many computational cost functions are similar plagued, by cost function noise, numerical intractability, or similar. Thus, the development of effective derivative-free optimization algorithms is of great importance to many fields of significance.

Compared to the sophisticated and highly effective derivative-based methods of optimization in widespread use today, the development of derivative-free optimization has generally been lacking. However, with increasing computing power comes a

greater number of optimization problems, many of which are ill-suited to derivative-based methods, making the study and implementation of derivative-free optimization methods paramount.

With this motivation as background, we delve into the theory of modern derivative-free optimization. Broadly speaking, the field can be divided into two subfields: global optimization, where the cost function is nonconvex and many local minima exists, and local optimization, where only a single optimum exists. Pedagogical analysis often emphasises the latter; however, in many problems relevant to society at large, cost functions tend to be nonconvex, constrained, nonsmooth, and otherwise challenging. When constrained appropriately, many nonconvex problems can be broken down into a series of convex local minima, therefore, both subfields must be considered.

Many global, or at least nonlocal, optimization algorithms have been proposed and implemented, with varying degrees of success. Perhaps the best known is Simulated Annealing, a process in which a stochastic search is allowed to converge according to an 'annealing schedule' varying the locality of the search, from coarse to fine. Motivated by the metallurgical process of annealing, Simulated Annealing is a commonly used nonlocal algorithm. Despite its popularity, SA algorithms tend to be only nonlocally, rather than globally, convergent, and the class of algorithms is generally inefficient with respect to function evaluations. That is, to reach a given level of convergence requires significantly more function evaluations than competing algorithms. This limits the utility of SA algorithms to the optimization of comparatively inexpensive cost functions or cost functions ill suited to other algorithms (e.g. discrete optimization).

Another well-known nonlocal class of algorithms is known as Genetic Algorithms, where each point in parameter space is named an 'individual', 'member', or 'genome'. Via a process of heuristics, the 'fittest' of the individuals are allowed to 'breed'. Via the introduction of a stochastic 'mutation', a number of 'offspring' points are generated, upon which the cost function is evaluated. Then the fittest individuals are again selected and allowed to breed. This process generates a sort of stochastic search. Unfortunately, Genetic Algorithms have virtually no theoretical underpinnings, convergence criteria are unknown, and their performance is generally very poorly understood. Somewhat humorously, many have struggled to provide a theoretical explanation of why

genetic algorithms perform as well as they do.

With these limitations in mind, Generalized Pattern Search (GPS) methods have been generally accepted as the most rigorous global and local derivative-free optimization methods yet devised. All GPS methods discretize the parameter space via a mesh or a grid, upon which all function evaluations are made. The optimization starts with a comparatively coarse mesh; as convergence is neared, the mesh is refined. This has the effect of keeping function evaluations far apart before convergence is attained, avoiding extensive exploration of a fruitless region. For nonlocal optimization problems, the Surrogate Management Framework (SMF) family of algorithms is an obvious choice. By melding a pattern search step offering efficient local optimization with a globally convergent Search step, SMF algorithms offer good local convergence *and* good global convergence, rather than sacrificing one for the other as SA and Genetic algorithms do. Both the pattern search step and the Search step of an SMF algorithm can be any of a great number of choices. However, all subalgorithms of SMF lie on a grid - and heretofore the only mesh choice examined has been the simple Cartesian grid.

In the examining of the efficiency of grid-based derivative free optimization algorithms, the authors have turned their attention to the underlying grid, in addition to the subalgorithms that SMF is built off of. Cartesian grids are only one, highly inefficient, grids, or lattices, that discretize parameter space. The mathematically mature, deep, and subtle field of n-dimensional sphere packings offers a wide range of very efficient lattices that compete directly with Cartesian grids.

Despite being elegant, mature, and extremely useful, engineering and science applications of this mathematical field have been limited. As such, a careful review of lattice theory, with emphasis on the tools necessary to apply this powerful body of literature to modern science and engineering challenges as well as a thorough review of previous engineering applications, particularly coding theory, can be found in Chapter 2. Included in Chapter 2 are a large number of previously unknown uniformity metrics calculated by the authors, as well as several new packings designed specifically for engineering applications. Many pertinent algorithms are clearly and succinctly presented to facilitate future application of this body of literature.

With the theoretical foundation of sphere packings presented in Chapter 2, Chap-

ter 3 delves into a lattice-based, globally convergent optimization algorithm named LABDOGS. Building the algorithm around a Successive Polling (SP) pattern search based on any of a large number of highly efficient lattices introduced in Chapter 2, the SP algorithm is introduced and thoroughly tested, demonstrating a significant increase in convergence rates compared to the analogous algorithm implemented on the Cartesian grid. With the SP Poll step defined and tested, Chapter 3 then introduces the subject of Kriging interplating functions, exploring globally convergent Search algorithms based on the Kriging structure, ending up with the most modern and efficient solution of maximizing the likelyhood of improvement (MLI). Combining the lattice-based SP Poll with the MLI Search step, the most efficient SMF-like algorithm to date, LABDOGS, is fully defined. Testing on a variety of cost functions verifies LABDOGS' superior convergence rates to Cartesian grid-based alternatives.

The LABDOGS algorithm works very well for global optimization problems, and allows for linear constraints on the cost function. The weakness of LABDOGS, as well as any other SP-based algorithm, is that the Poll step algorithm fails to converge under particular conditions. In particular, when hard constraints are encountered, the SP algorithm generally fails to locate a feasible descent direction and fails to converge. In LABDOGS, this does not prevent the overall algorithm convergence; however, it is desirable to implement a superior Poll step selected from the Mesh Adaptive Direct Search (MADS) class of algorithms.

MADS algorithms are a sophisticated subclass of pattern searches that allow for the polling directions to become dense as the optimization progresses. By not reorienting the poll step when a successful poll has been realized, a good MADS algorithm will perform analogously to a line search when challenging cost function behaviors are encountered. The best algorithm of the MADS subclass is known as OrthoMADS. Defined on a Cartesian grid, OrthoMADS demonstrates some shortcomings in convergence rates, if not in convergence behaviors. In Chapter 3 OrthoMADS is discussed and analyzed. By utilizing the n-dimensional lattices from Chapter 1 in a MADS framework, the Λ-MADS algorithm is defined and each component of the algorithm is thoroughly tested. Λ-MADS maintains all the desirable convergence characteristics of OrthoMADS, but offers improved convergence rates; in $n = 8$ approximately half as many function eval-

uations are required to converge compared to OrthoMADS. This improvement in convergence is the result of the low quantization error of the lattice, the greater radius uniformity of the poll set, the minimal rather than maximal positive basis, and several other distinguishing changes from OrthoMADS. The testing presented in Chapter 3 clearly establishes Λ-MADS as the most efficient MADS algorithm yet devised.

The material covered in Chapter 1 is leveraged heavily throughout Chapters 2 and 3, to great effect, firmly establishing the benefits of utilizing efficient lattices to discretize parameter space. Combined with the results in Bewley & Cessna (2011), this sequence of work demonstrates the potential impact that widespread application of *n*-dimensional sphere packings may have in modern science and engineering applications.

Chapter 2

New Horizons in Sphere Packing Theory, Part I: Fundamental Concepts and Constructions, from Dense to Rare

2.1 Introduction

The field of n-dimensional sphere packings is elegant and mature in its mathematical development and characterization. However, it is still relatively limited in its practical applications, especially for $n > 3$. The present line of research intends to open up two broad new areas for profitable application of this powerful body of mathematical literature in science and engineering. Towards this end, the present work (Chapter 2) reviews the essential results available in this field (reconciling the theoretical literature for dense and rare sphere packings, which today are largely disjoint), catalogs the key properties of the principle dense and rare sphere packings and corresponding nets available (including hundreds of values not previously known), and extends the study of regular rare sphere packings and nets to $n > 3$ dimensions (an area which up to now has been largely unexplored). These results are leveraged heavily in the practical applications addressed in Chapers 3 and 4. In particular, Chapter 3 builds from this presentation to develop a new algorithm for Lattice-Based Derivative-free Optimization via Global Surrogates (LABDOGS), leveraging dense sphere packings as an alternative

to Cartesian grids to coordinate derivative-free searches; Chapter 3 also develops and uses a new algorithm for efficient solution of discrete Thompson problems restricted to nearest-neighbor points of a lattice. Chapter 4 builds off the material in Chapters 2 and 3 to develop a new, highly efficient Mesh Adaptive Direct Search (MADS) optimization algorithm named Λ-MADS. The introduction of lattices allows Λ-MADS to converge more than twice as efficiently as the current best MADS algorithm. The present sequence of research projects establishes that significant performance improvements may be realized by leveraging *n*-dimensional sphere packings appropriately in such practical applications.

2.2 An introduction to lattices

An *n*-dimensional infinite *sphere packing* is an array of nodal points in $\mathbb{R}^n$ obtained via the packing of identical *n*-dimensional spheres. By *packing*, we mean an equilibrium configuration of spheres, each with at least 2 nearest neighbors, against which a repellant force is applied. Many packings investigated in the literature are *stable* packings, meaning that there is a restoring force associated with any small movement of any node of the packing; this requires each sphere in the (*n*-dimensional) packing to have at least $n+1$ neighbors. However, unstable packings with lower nearest-neighbor counts are also of interest. Note also that, by replacing each sphere in an *n*-dimensional packing with a nodal point (representing, e.g., a computer), and connecting those nodal points which are nearest neighbors, a *net* (a.k.a. *interconnect* or *contact graph*) is formed[1].

An *n*-dimensional real *lattice* (a.k.a. *lattice packing*) is a sphere packing which is shift invariant (that is, which looks identical upon shifting any nodal point to the origin); this shift invariance generally makes lattice packings simpler to describe and enumerate than their nonlattice alternatives. Note that there are many regular[2] sphere packings which are *not* shift invariant [the nonlattice packings corresponding to the honeycomb net in 2D and the diamond and quartz nets in 3D are some well-known examples]. We will focus our attention in this work on those packings and nets which

[1] As introduced in the second-to-last paragraph of §2.3.3, it is natural with certain sphere packings (for example, D_n^*, A_n^r, and the packings associated with the T_n^{90} and T_n^{60} nets) to define nets which are *not* contact graphs of the corresponding sphere packings by connecting non-nearest-neighbor points.

[2] The regularity of a nonlattice packing is quantified precisely in §2.4.1.

are at least *uninodal* (that is, which look identical upon shifting any nodal point to the origin and rotating and reflecting appropriately). For *dense* sphere packings, from a practical perspective, lattice packings are essentially [3] as good a choice as their more cumbersome nonlattice alternatives for $n \leq 24$ in terms of the four metrics defined below (that is, for maximizing packing density and kissing number and minimizing covering thickness and quantization error). However, the best *rare* sphere packings (with small kissing number) are all nonlattice packings.

As illustrated in Figure 2.1 and Table 2.1, we may introduce the subject of n-dimensional sphere packings by focusing our attention first on the $n = 2$ case: specifically, on the *triangular*[4] lattice (A_2), the *square* lattice ($\mathbb{Z}^2$), and the *honeycomb* nonlattice packing (A_2^+). The characteristics of such sphere packings may be quantified by the following measures:

• The *packing radius* of a packing, ρ, is the maximal radius of the spheres in a set of identical nonoverlapping spheres centered at each nodal point.

• The *packing density* of a packing, Δ, is the fraction of the volume of the domain included within a set of identical non-overlapping spheres of radius ρ centered at each nodal point on the packing. Packings that maximize this metric are referred to as *close-packed*.

• The *covering radius* of a packing, R, is the maximum distance between any point in the domain and its nearest nodal point on the packing. The *deep holes* of a packing are those points which are at a distance R from all of their nearest neighbors. Typical vectors from a nodal point to the nearest deep holes in a lattice packing are often denoted [1], [2], etc.

• The *covering thickness* of a packing, Θ, is the number of spheres of radius R centered at each nodal point containing an arbitrary point in the domain, averaged over the domain.

• The *Voronoï cell* of a nodal point in a packing, $\Omega(P_i)$, consists of all points in the

[3] For $n = 10$, 11, 13, 18, 20, and 22, there exist nonlattice packings (denoted P_{10c}, P_{11a}, P_{13a}, $\mathscr{B}_{18}^*$, $\mathscr{B}_{20}^*$, $\mathscr{A}_{22}^*$) that are 8.3%, 9.6%, 9.6%, 4.0%, 5.2%, and 15.2% denser then the corresponding best known lattice packings (Conway & Sloane 1999, p. xix); to put this into perspective, the density of Λ_{22} is over 10^6 *times* the density of $\mathbb{Z}^{22}$.

[4] Note that many in this field refer to the A_2 lattice as "hexagonal". We prefer the unambiguous name "triangular" to avoid confusion with the honeycomb nonlattice packing (see Figure 2.1).

domain that are at least as close to the nodal point P_i as they are to any other nodal point P_j.

• The *mean squared quantization error per dimension* of a lattice or uninodal nonlattice packing, G, is the average mean square distance of any point in the domain to its nearest nodal point, normalized by n times the appropriate power of the volume, V, of the Voronoï cell. Shifting the origin to be at the centroid of a Voronoï cell $\Omega(P_i)$, it is given by

$$G = \frac{S}{nV^{\frac{n+2}{n}}} \quad \text{where} \quad S = \int_{\Omega(P_i)} |\mathbf{x}|^2 \, d\mathbf{x}, \quad V \int_{\Omega(P_i)} d\mathbf{x}. \tag{2.1}$$

• The *kissing number* of a lattice or uninodal nonlattice packing, τ, is the number of nearest neighbors to any given nodal point in the packing. In other words, it is the number of spheres of radius ρ centered at the nodal points of the packing that touch, or "kiss", the sphere of radius ρ centered at the origin.

• The *coordination number* of a net (derived from a sphere packing, as discussed previously) is the first number of the net's *coordination sequence*, the k'th element of which is given by $td_k - td_{k-1}$, where td_k, which quantifies the net's *local topological density*, is the total number of nodes reached via k hops or less from the origin in the net[5].

Certain applications, such as that explored in Chapter 3 of this work (Belitz & Bewley 2011), require dense lattices. There are two key drawbacks with cubic approaches for such applications. First, the *discretization of space is significantly less uniform* when using the cubic grid as opposed to the available alternatives, as measured by the packing density Δ, the covering thickness Θ, and the mean-squared quantization error per dimension, G (see Table 2.1). Second, the *configuration of nearest-neighbor gridpoints is significantly more limited* when using the cubic grid, as measured by the kissing number τ, which is an indicator of the degree of flexibility available when selecting from nearest-neighbor points. As seen by comparing the $n = 2$, $n = 8$, and $n = 24$

[5] In most cases, the natural net to form from a sphere packing is the contact graph; in such cases, the kissing number, τ, and the coordination number are equal. As mentioned previously, it is natural with certain sphere packings to define nets which are *not* contact graphs by connecting non-nearest-neighbor points; in such cases, the kissing number (a property of the sphere packing) and the coordination number (as defined here, a property of a corresponding net) are, in general, *not* equal. We find this clear semantical distinction to be useful to prevent confusion between these two distinct concepts; note that some authors (e.g., Conway & Sloane 1999) do not make this distinction.

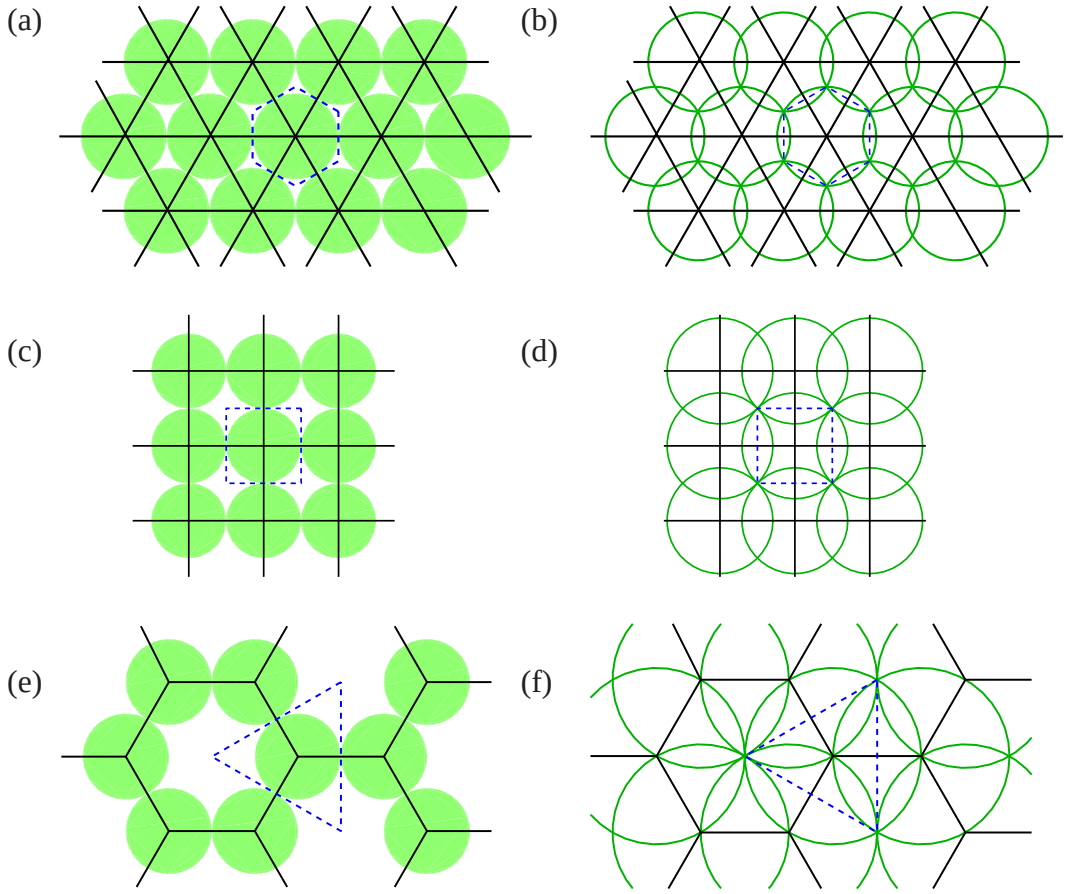

Figure 2.1: The triangular lattice (a,b), the square lattice (c,d), and the honeycomb nonlattice packing (e,f). Indicated in the left three subfigures is the *packing* with spheres of radius ρ, the corresponding *net* or *contact graph* (solid lines), a typical *Voronoï cell* (dashed line), and the *kissing number* (that is, the spheres that contact a given sphere). Indicated in the right three subfigures is the *covering* with spheres of radius R. Looking at their respective packing densities Δ in Table 2.1, as compared with the square lattice, the triangular lattice is said to be **dense**, and the honeycomb nonlattice packing is said to be **rare**.

cases in Table 2.1, these drawbacks become increasingly substantial as the dimension n is increased; by the dimension $n = 24$, the cubic grid has

- a factor of $0.001930/1.1501e-10 \approx 17,000,000$ worse (lower) packing density,

Table 2.1: Characteristics of selected lattice and uninodal nonlattice packings and nets.

n	**packing**	**name**	Δ	Θ	G	τ
2	A_2	triangular	0.9069	1.2092	0.08019	6
	$\mathbb{Z}^2$	square	0.7854	1.5708	0.08333	4
	A_2^+	honeycomb	0.6046	2.4184	0.09623	3
8	E_8	Gosset	0.2537	4.059	0.07168	240
	$\mathbb{Z}^8$	cubic	0.01585	64.94	0.08333	16
	(unstable)	V_8^{90}	5.590e-4	49.89	0.09206	4
		Y_8^{90}	2.327e-4	87.31	0.09266	3
24	Λ_{24}	Leech	0.001930	7.904	0.06577	196560
	$\mathbb{Z}^{24}$	cubic	1.150e-10	4,200,263	0.08333	48

- a factor of $4,200,263/7.9035 \approx 530,000$ worse (higher) covering thickness,
- a factor of $0.08333/0.0658 \approx 1.27$ worse (higher) mean-squared quantization error, and
- a factor of $196560/48 \approx 4100$ worse (lower) kissing number

than the densest available alternative lattice. Thus, the selection of the cubic grid, by default, for applications requiring dense (that is, uniform) lattices with $n > 3$ is simply untenable.

Other applications, such as that explored in Cessna & Bewley (2011), require regular nets which, with low coordination number, connect to a large number of nodes with each successive hop from the origin, as quantified by the net's coordination sequence. As mentioned previously, a useful measure of a net's topological density is given, e.g., by td_{10}, which is the number of distinct nodes within 10 hops of the origin. Note that the coordination number of the n-dimensional cubic grid is $2n$; the coordination number of the alternative n-dimensional constructions introduced in §2.4 are as small as 3 or 4, while the topological density increases rapidly as n is increased (compare, e.g., the values of td_{10} for A_2^+ and $\mathbb{Z}^2$, with $\tau = 3$ and $\tau = 4$ respectively, to those for Y_8^{90} and V_8^{90} in Table 2.1); it is thus seen that, for applications requiring graphs with

low coordination number and high topological density, the selection of the cubic grid, by default, is also untenable.

We are thus motivated to make the fundamental results of both dense and rare n-dimensional sphere packing theory more broadly accessible to the science and engineering community, and to illustrate how this powerful body of theory may be put to use in two important new applications of practical relevance. Towards this end, the remainder of Chapter 2 succinctly reviews and extends several significant results in this mature and sophisticated field, inter-relating the literature on dense and rare packings, which is today largely disjoint. These results are leveraged heavily in the applications described in Chapter 3 and 4. We note that, beyond providing an up-to-date and synthetic review of this otherwise difficult subject in a (hopefully) accessible language, a significant number of new computations, constructions, algorithms, and metrics are also reported in Chapter 2 [the reader is referred specifically to Tables 2.2-2.3, §2.4.4 through §2.4.4, and §2.4.5].

The mathematical characterization of sphere packings has a long and rich history. Some recent articles and popular books recount this history in detail, including Zong (1999), Szpiro (2003), Hales (2006), and Aste & Weaire (2008). The purpose of the present article is not to repeat these historical retrospectives, which these sources do quite adequately, but to characterize, catalog, and extend the infinite packings available today to facilitate their practical application in new fields. Nonetheless, we would remiss if we didn't at least provide a brief historical context to this field, which we attempt in the following two subsections.

2.2.1 Finite packings: mystic marbles, stacked spheres, permuted planets, cartoned cans, catastrophic sausages, and concealed origins

We begin by defining, for $m \geq 1$, a notation to build from:

$$T_{0,m} \triangleq 1, \qquad T_{1,m} \triangleq \sum_{k=1}^{m} T_{0,k} = m \ \text{ (the positive integers).}$$

In the sixth century BC, Pythagoras and his secret society of numerologists, the Pythagoreans, discovered geometrically (see Figure 2.2, and pp. 43-50 of Heath 1931) the formula

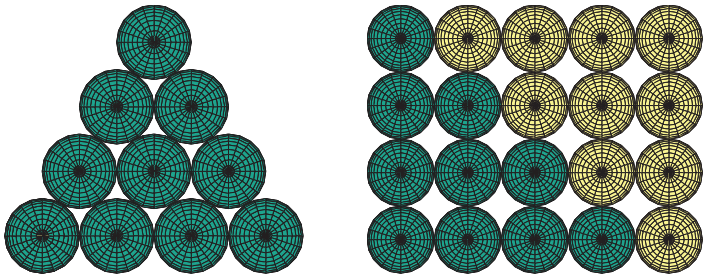

Figure 2.2: Ten marbles placed in a triangle (left) [referred to by the Pythagoreans as a $\tau\varepsilon\tau\rho\alpha\kappa\tau\acute{\upsilon}\varsigma$, (tetractys), and upon which they placed a particular mystic significance], and (right) the Pythagoreans' placement of two triangular groups of marbles into an "oblong" $m \times (m+1)$ rectangle, from which the formula for $T_{2,m}$ follows immediately.

for the number of marbles placed in a (2D) triangle (that is, the "triangular numbers"):

$$T_{2,m} \triangleq \sum_{k=1}^{m} T_{1,k} = m(m+1)/2.$$

The earliest known mathematical work to discuss the (3D) stacking of objects is a Sanskrit document *The Aryabhatiya of Aryabhata* (499 AD; see Clark 1930, p. 37), which states:

"In the case of an *upaciti* [lit., 'pile'] which has ... the product of three terms, having the number of terms for the first term and one as the common difference, divided by six, is the *citighana* [lit., 'cubic contents of the pile']. Or, the cube of the number of terms plus one, minus the cube root of this cube, divided by six."

Thus, Aryabhata establishes, in words, two equivalent expressions for the number of objects ("cubic contents") in a (3D) triangular-based pyramid ("pile") with m objects on each edge:

$$T_{3,m} = \frac{m(m+1)(m+2)}{3!} = \frac{(m+1)^3 - (m+1)}{6};$$

note also that $T_{3,m} \triangleq \sum_{k=1}^{m} T_{2,k}$.

Thomas Harriot was apparently the first to frame the problem of sphere packing mathematically in modern times (see, e.g., the biography of Harriot by Rukeyser 1972). At the request of Sir Walter Raleigh, for whom Harriot served, among other capacities, as an instructor of astronomical navigational and on various problems related to gunnery, Harriot (on December 12, 1591) computed, but did not publish, the number of

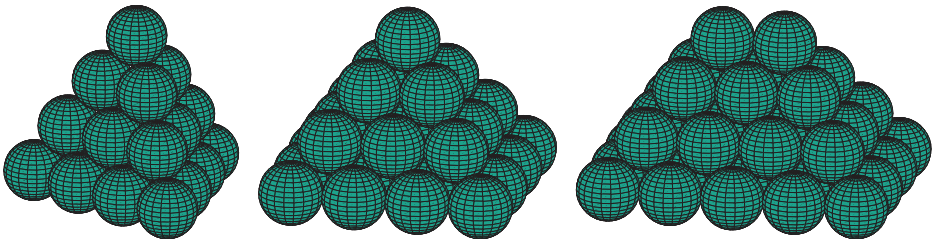

Figure 2.3: Pyramidal stacks of spheres with triangular, square, and "oblong" (rectangular) bases. All three stacks are subsets of the face-centered cubic lattice, discussed further in §2.3.3.

cannonballs in a pile with a triangular, square $[m \times m]$, and rectangular $[m \times (m+1)$, a.k.a. "oblong"] base, as illustrated in Figure 2.3, obtaining $T_{3,m}$, S_m, and R_m respectively, where

$$S_m = \sum_{k=1}^{m} k^2 = \frac{m(m+1)(2m+1)}{6}, \quad R_m = \sum_{k=1}^{m} k(k+1) = S_m + T_{2,m} = \frac{m(m+1)(2m+4)}{6}.$$

In 1614, Harriot wrote *De Numeris Triangularibus Et inde De Progressionibus Artithmeticis: Magisteria magna* (*On triangular numbers and thence on arithmetic progressions: the great doctrine*)[6]. Looking closely at the triangular table of binomial coefficients[7] on pp. 1-3 (folios 108-110) of this remarkable document, it is seen that Harriot understood the *geometric* relationship between the positive integers $T_{1,m}$, the "triangular numbers" $T_{2,m}$ [that is, the number of spheres in a (2D) triangle with m spheres on each edge], the "pyramidal numbers" $T_{3,m}$ [that is, the number of spheres in a (3D) trianglar-based pyramid with m spheres on each edge], and the next logical steps in this arithmetic progression, given by:

$$T_{4,m} \triangleq \sum_{k=1}^{m} T_{3,k} = \frac{m(m+1)(m+2)(m+3)}{4!}$$

$$T_{5,m} \triangleq \sum_{k=1}^{m} T_{4,k} = \frac{m(m+1)(m+2)(m+3)(m+4)}{5!},$$

[6] Harriot (1614) passed through several hands before finally being published in 2009, almost 4 centuries later.

[7] This now famous triangular table of binomial coefficients is incorrectly attributed by many in the west to Blaise Pascal (b. 1623), though it dates back to several earlier sources, the earliest being Pingala's Sanskrit work *Chandas Shastra*, written in the fifth century BC.

etc. In particular, Harriot noticed that the $(n+1)$'th element of the $(n+m)$'th row of this triangular table is $T_{n,m}$. Accordingly, we may think of $T_{n,m}$ as the number of spheres in an "n-dimensional pyramid" with m spheres on each edge, with $T_{n,2}$ representing $n+1$ spheres configured at the corners of an n-dimensional simplex. It is thus natural to credit Harriot (1614) with the first important steps towards the discovery of laminated lattices, discussed further in §2.3.4 and §2.3.6.

Harriot also introduced the packing problem to Johannes Kepler, ultimately leading Kepler (1611), in another remarkable document *Strena seu de nive sexangula* (*The six-cornered snowflake*), which also hypothesized about a related atomistic physical basis for hexagonal symmetry in crystal structures of water, to conjecture that

"The (cubic or hexagonal close) packing is the tightest possible, such that in no other arrangement can more spheres be packed into the same container."

Kepler's conjecture is, of course, patently false if considered in a finite container of a specified shape. For instance, a $2d \times 2d \times 2d$ cubic container can fit 8 spheres of diameter d if arranged in Cartesian configuration, but can only fit 5 spheres if arranged in a "close-packed" configuration[8]. It is presumed that Kepler in fact recognized this, and thus Kepler's conjecture is commonly understood as a conjecture regarding the densest packing possible in the limit that the size of the container is taken to infinity (for further discusssion, see §2.2.2).

Note in Figure 2.3 that any sphere (referred to as a "sun") on the interior of the piles has 12 nearest neighbors (referred to as its "planets"). Considering this sun and its 12 planets in isolation, there is in fact adequate room to permute the planets to different positions while keeping them in contact with the sun, something like a 12-cornered Rubik's cube with spherical pieces (see Figure 2.4). Due to the extra space available in this configuration, it is unclear upon first inspection whether or not there is sufficient room to fit a 13'th planet in to touch the sun while keeping all of the other 12 planets in contact with it. In 1694, Isaac Newton conjectured this could not be done, in a famous disagreement with David Gregory, who thought it could. Newton turned out to be right, with a complete proof first given in Schütte & van der Waerden (1953), and a substantially simplified proof given in Leech (1956).

Moving from 16th-century stacks of cannonballs to 21st-century commerce, the

[8]For larger containers, the arrangements which pack in the greatest number of spheres (or other objects) must in general be found numerically (see Gensane 2004, Schürmann 2006, and Friedman 2009).

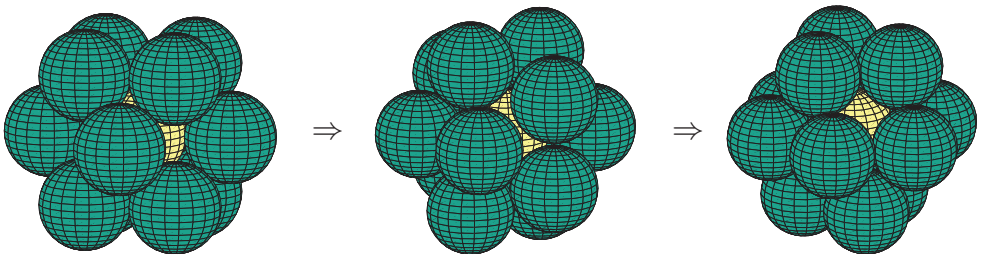

Figure 2.4: Illustration of the 13 spheres (a.k.a. Newton-Gregory) problem and planetary permutations. Configuration (a) is 13 of the spheres taken from the second, third, and fourth layers of the stack in the orientation shown in Figure 2.3b, whereas configuration (c) is 13 of the spheres taken from the third, fourth, and fifth layers of the stack in the orientation shown in Figure 2.3a [extended by one additional layer]. In both configurations, the 12 "planets" (positioned around the central "sun") are centered at the vertices of a cuboctahedron. The planets can be permuted by "pinching" together two of the four planets on the corners of each square face, in an alternating fashion, to form a symmetric icosahedral configuration with significant space between each pair of planets [configuration (b)], then "pushing" apart pairs of planets in an analogous fashion to form a different cuboctahedron. Alternatively, starting from configuration (b), identifying any pair of opposite planets as "poles", and slightly shifting the five planets in each of the "tropics" as close as possible to their nearest respective poles, the resulting northern and southern groupings of planets can be rotated in relation to each other along the equator. Repeated application of these two fundamental motions can be used to permute the planets arbitrarily.

question of dense finite packings of circles and spheres finds practical relevance in a variety of packaging problems. For example, to form a rectangular cardboard carton for 12 fl oz soda cans, 164 cm^2 of cardboard per can is needed if 18 cans are placed in a cartesian configuration with 3 rows of 6 cans per row, whereas 3.3% less cardboard per can is needed if 18 cans are placed in a triangular configuration (within a rectangular box) with 5 rows of {4,3,4,3,4} cans per row. If an eye-catching (stackable, strong, "green"...) hexagonal cardboard carton for the soda cans is used, with 19 cans (described in marketing terms as "18 plus 1 free") again placed in a triangular configuration, 17.7% less cardboard per can is required.

Two new questions arise when one "shrink-wraps" a number (m) of n-dimensional spheres (resulting in a convex, fitted container), namely: what configuration of the spheres minimizes the surface area of the resulting container, and what configuration minimizes the volume of the resulting container? Both questions remain open, and are reviewed in Zong (1999). Regarding the minimim surface area question, it was conjectured by Croft, Falconer, & Guy (1991) that the minimum surface area, for $n \geq 2$ and large m, is achieved with a roughly spherical arrangement. In contrast, regarding the minimim volume question, it was conjectured by L. Fejes Tóth (1975) that the minimum volume, for $n \geq 5$ and any m, is achieved by placing the spheres in a line, leading to a shrink-wrapped container in the shape of a "sausage". For $n = 3$, it has been shown that a roughly spherical arrangement minimizes the volume for $m = 56$, $m = 59$ to 62, and $m \geq 65$, and it is conjectured that a sausage configuration minimizes the volume for all other m (see Gandini & Willis 1992); for $n = 4$, there appears to be a similar "catastrophe" in the volume-minimizing solution, from a sausage configuration to a roughly spherical configuration, as m is increased beyond a critical value (Willis 1983 conjectures this critical value to be $m \approx 75000$, whereas Gandini & Zucco 1992 conjectures it to be $m = 375769$).

Finally, L. Fejes Tóth (1959) presents a curious set of questions that arise when considering the blocking of light with a finite number of opaque unit spheres packed around the origin. The first such question, known as Hornich's Problem, seeks the smallest number of opaque unit spheres that completely conceal light rays emanating from a point source at the center of a transparent unit sphere at the origin. A related question, known as L. Fejes Tóth's Problem, seeks the smallest number of opaque spheres that completely conceal light rays emanating from the surface of a unit sphere at the origin (e.g., in Figure 2.4, adding additional outer planets to completely conceal the view of the sun from all angles). In 2D, the (trivial) answer to both problems is 6, via the triangular packing indicated in Figure 2.1a. In higher dimensions, both questions remain open, and the answer differs depending on whether or not the sphere centers are restricted to the nodal points of a lattice. For the L. Fejes Tóth's Problem, for $n \geq 3$, the answer is unbounded if restricted to lattice points, and bounded if not. For Hornich's Problem, the answer is bounded in both cases, with the number of spheres, h, required in the 3D case,

when not restricted to lattice points, being somewhere in the range $30 \leq h \leq 42$. Zong (1999) derives several of the known bounds available in both problems.

2.2.2 Infinite packings

In the last 300 years, *many* different constructions of infinite lattice and nonlattice packings have been proposed in each dimension. These packings each have different packing density, covering thickness, mean-squared quantization error, and kissing number, and their corresponding nets each have different topological density; knowledge of these properties is essential when selecting a packing or net for any given application. We have thus attempted to catalog these constructions and their properties thoroughly in the remainder of this review.

In the characterization of density, amongst all *lattice* packings of a given dimension, the A_2, A_3, D_4, D_5, E_6, E_7, E_8, and Λ_{24} constructions given in §2.3 have been proven to be of maximum density, in Lagrange (1773) for $n = 2$, Gauss (1831) for $n = 3$, Korkine & Zolotareff (1873, 1877) for $n = 4$ and 5, Blichfeldt (1935) for $n = 6$ through 8, and Cohn & Kumar (2009) for $n = 24$. There are no such proofs of optimality for other values of n, though the lattices Λ_n and K_n introduced in §2.3.6 are likely candidates in the range $9 \leq n \leq 23$.

Remarkably, if one considers both lattice *and* nonlattice packings, proof of which packing is of maximum density in a given dimension is still open for $n > 3$. It was established in Thue (1892) that A_2 has the maximum density amongst all lattice and nonlattice packings for $n = 2$. Considerable attention has been focused over the centuries on the corresponding question for A_3 in dimension $n = 3$, that is, on Kepler's conjecture (posed in 1611) in the limit that the container size is taken to infinity. Indeed, David Hilbert, in his celebrated list of 23 significant open problems in mathematics in 1900, included a generalization of Kepler's conjecture as part of his 18th problem (see, e.g., Milnor 1976).

Note that it is not at all obvious that an infinite packing as regular as A_3 would necessarily be the packing that maximizes density. Indeed, as mentioned in footnote 3 on page 10, nonlattice packings are known in dimensions $n = 10$, 11, 13, 18, 20, and 22 that are each slightly denser than the densest known lattice packings in these dimensions.

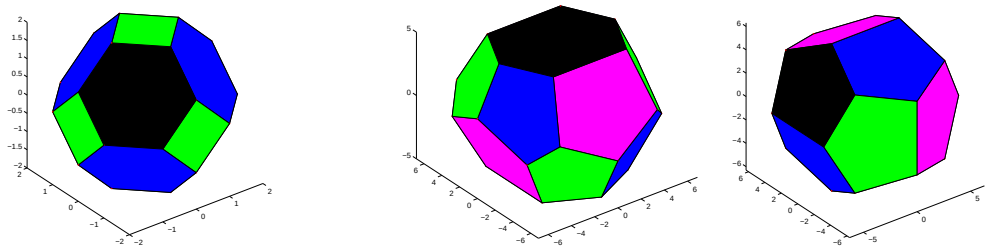

Figure 2.5: A regular truncated octahedron (a), used to tile $\mathbb{R}^3$ in Kelvin's conjecture; (b) an irregular tetrakaidecahedron and dodecahedron, used to tile $\mathbb{R}^3$ in the Weaire-Phelan structure.

In three dimensions, physiologist Stephen Hales (1727), in his groundbreaking work *Vegetable Staticks*, reported a curious experiment:

"I compressed several fresh parcels of Pease in the same Pot, ... by the great incumbent of weight, pressed into the interstices of the Pease, which they adequately filled up, being therefore formed into pretty regular dodecahedrons."

This report implied that many of the dilated peas in this experiment had 12 nearest neighbors and/or pentagonal faces. However, the "pretty regular" qualification left a certain ambiguity, and this experiment left mathematicians puzzled, as it is patently impossible to tile $\mathbb{R}^3$ with regular dodecahedra. Kelvin (1887) formalized the question inherent in Hales' dilated pea experiment by asking how $\mathbb{R}^3$ could be divided into regions of equal volume while minimizing the partitional area. He conjectured the answer to be a regular tiling of $\mathbb{R}^3$ with truncated octahedra, which are in fact the Voronoï cells of the A_3^* lattice (see §2.4.4). [Note that the Voronoï cell of the A_3 lattice is the (face-transitive) *rhombic* dodecahedron, which is dual to the cuboctahedron illustrated in Figures 2.4a,c and tiles $\mathbb{R}^3$ with slightly greater partitional area than does the tiling with truncated octahedra.] Kelvin's conjecture stood for over 100 years, until Weaire & Phelan (1994) discovered a tiling of $\mathbb{R}^3$ based on irregular tetrakaidecahedra (with 2 hexagonal faces and 12 pentagonal faces) and irregular dodecahedra (with 12 pentagonal faces); this tiling has 0.3% less partitional area than the much more regular tiling with truncated octahedra considered by Kelvin (see Figure 2.5). In hindsight, it is quite possible that Hales might have in fact stumbled upon the Weaire-Phelan structure in his cooking pot (in 1727!) and, seeing all of those pentagonal faces and 12-sided (as well as 14-sided) dilated peas, as-

serted that what he was looking at was a culinary approximation to a tiling of $\mathbb{R}^3$ with regular dodecahedra, even though such a tiling is impossible.

Returning to Kepler's conjecture, in 1998, Thomas Hales (no relation to Stephen) announced a long-sought-after proof, in a remarkably difficult analysis making extensive use of computer calculations. This proof was spread over a sequence of papers published in the years that followed (see Hales 2005). An extensive discussion of this proof, which is still under mathematical scrutiny, is given in Szpiro (2003). Inspiration for this proof was based, in part, on a strategy to prove Kepler's conjecture proposed by L. Fejes Tóth (1953), the first step of which is a quantitative version of the Newton-Gregory problem discussed in §2.2.1.

2.3 Dense lattice packings for $n \leq 24$

There are many dense lattices more complex than the cubic lattice that offer superior uniformity and nearest-neighbor configuration, as quantified by the standard metrics introduced in §2.2 (namely, packing density, covering thickness, mean-square quantization error, and kissing number). This section provides an overview of many of these lattices; *the definitive comprehensive reference for this subject is Conway & Sloane (1999), to which the reader is referred for much more detailed discussion and further references on most of the topics discussed in §2.3.* Note that the subject of coding theory, reviewed in §2.5, is very closely related to the subject of dense lattice packings. As mentioned in the abstract, the practical application explored in Chapters 3 and 4 of this work also leverages these constructions heavily.

2.3.1 Lattice terminology

The notation $L_n \cong M_n$ means that the lattices L_n and M_n are *equivalent* (when appropriately rotated and scaled) at the specified dimension n. Also note that the four most basic families of lattices introduced in §2.3, denoted $\mathbb{Z}^n$, A_n, D_n, and E_n, are often referred to as *root lattices* due to their relation to the root systems of Lie algebra.

There are three primary methods[9] to define any given n-dimensional real lattice:

[9] A convenient alternative method for building a cloud of lattice points near the origin is based on the

- As an *explicit description* of the points included in the lattice.
- As an *integer linear combination* (that is, a linear combination with integer coefficients) of a set of n basis vectors $\mathbf{b}^i$ defined in $\mathbb{R}^{n+m}$ for $m \geq 0$; for convenience, we arrange these basis vectors as the columns[10] of a *basis matrix*[11] B.
- As a *union of cosets*, or sets of nodal points, which themselves may or may not be lattices.

The standard form of these definitions, as used in §2.3, makes it straightforward to generalize application codes that can build easily upon any of the lattices so described.

Note that any real (or complex) lattice L_n also has associated with it a *dual lattice*, denoted L_n^*, which is defined such that

$$L_n^* = \left\{ \mathbf{x} \in \mathbb{R}^n \text{ (or } \mathbb{C}^n) \,:\, \mathbf{x} \cdot \bar{\mathbf{u}} \in \mathbb{Z} \text{ for all } \mathbf{u} \in L_n \right\}, \tag{2.2}$$

where $\mathbb{Z}$ denotes the set of all integers, dot denotes the usual scalar product, and overbar denotes the usual complex conjugate. If B is a square basis matrix for L_n, then B^{-T} is a square basis matrix for L_n^*.

Unless specified otherwise, the word lattice in this work implies a real lattice, defined in $\mathbb{R}^n$. However, note that it is straightforward to extend this work to complex lattices, defined in $\mathbb{C}^n$. To accomplish this extension, it is necessary to extend the concept of the integers, which are used to construct a lattice via the "integer" linear combination of the basis vectors in a basis matrix B, as described above. There are two primary such extensions:

- The *Gaussian integers*, defined as $\mathscr{G} = \{a + b\imath \,:\, a, b \in \mathbb{Z}\}$ where $\imath = \sqrt{-1}$, which lie on a square array in the complex plane $\mathbb{C}$.

stencil of nearest-neighbor points to the origin in the lattice, repeatedly shifting this stencil to each of the lattice points near the origin determined thus far in order to create additional lattice points in the cloud. Unfortunately, this simple alternative method does not work for all lattices, such as D_n^* and A_n^r (see §2.3.3 and 2.3.4).

[10] In the literature on this subject, it is more common to use a *generator matrix M* to describe the construction of lattices. The basis matrix convention B used here is related simply to the corresponding generator matrix such that $B = M^T$; we find the basis matrix convention to be more natural in terms of its linear algebraic interpretation.

[11] Note that integer linear combinations of the columns of most matrices do *not* produce lattices (as defined in the second paragraph of §2.2). The matrices listed in §2.3 as basis matrices are special in this regard. Note also that basis matrices are not at all unique, but the lattices constructed from alternative forms of them are equivalent; the forms of the basis matrices listed in §2.3 were selected based on their simplicity.

- The *Eisenstein integers*, defined as $\mathscr{E} = \{a + b\omega : a, b \in \mathbb{Z}\}$ where $\omega = (-1 + \imath\sqrt{3})/2$ [note that $\omega^3 = 1$], which lie on a triangular array in the complex plane $\mathbb{C}$.

We may thus define three types of lattices from a basis matrix B:

- a real lattice, defined as a linear combination of the columns of B with integers as weights;
- a (complex) $\mathscr{G}$ lattice, defined as a linear combination of the columns of B with Gaussian integers as weights; and
- a (complex) $\mathscr{E}$ lattice, defined as a linear combination of the columns of B with Eisenstein integers as weights.

The special n-dimensional real, $\mathscr{G}$, and $\mathscr{E}$ lattices formed by taking $B = I_{n\times n}$ are denoted $\mathbb{Z}^n$, $\mathbb{Z}[\imath]^n$, and $\mathbb{Z}[\omega]^n$ respectively. Note also that, for any complex lattice with elements $\tilde{\mathbf{z}} \in \mathbb{C}^n$, there is a corresponding real lattice with elements $\tilde{\mathbf{x}} \in \mathbb{R}^{2n}$ such that

$$\tilde{\mathbf{x}} = \begin{pmatrix} \Re\{\tilde{z}_1\} & \Im\{\tilde{z}_1\} & \dots & \Re\{\tilde{z}_n\} & \Im\{\tilde{z}_n\} \end{pmatrix}^T. \tag{2.3}$$

The present work focuses on the practical use of real lattice and nonlattice packings with $n > 3$. Thus, in the present work, we only make brief use of complex lattices to simplify certain constructions.

2.3.2 The cubic lattice $\mathbb{Z}^n$

The *cubic lattice*, $\mathbb{Z}^n$, is defined $\mathbb{Z}^n = \{(x_1, \dots, x_n) : x_i \in \mathbb{Z}\}$, and may be constructed via integer linear combination of the columns of the basis matrix $B = I_{n\times n}$. The cubic lattice is self dual [that is, $(\mathbb{Z}^n)^* \cong \mathbb{Z}^n$] for all n.

2.3.3 The checkerboard lattice D_n, its dual D_n^*, and the offset checkerboard packing D_n^+

The *checkerboard lattice*, D_n, is an n-dimensional extension of the 3-dimensional *face-centered cubic* (*FCC*, a.k.a. *cubic close packed*) lattice. It is defined

$$D_n = \{(x_1, \dots, x_n) \in \mathbb{Z}^n : x_1 + \dots + x_n = \text{even}\}, \tag{2.4a}$$

and may be constructed via integer linear combination of the columns of the $n \times n$ basis matrix

$$B_{D_n} = \begin{pmatrix} -1 & 1 & & & 0 \\ -1 & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ 0 & & & & -1 \end{pmatrix}. \tag{2.4b}$$

The dual of the checkerboard lattice, denoted D_n^* and reasonably identified as the *offset cubic lattice,* is an n-dimensional extension of the 3-dimensional *body-centered cubic* (*BCC*) lattice. It may be written as

$$D_n^* = D_n \cup ([1] + D_n) \cup ([2] + D_n) \cup ([3] + D_n) \cong \mathbb{Z}^n \cup ([1] + \mathbb{Z}^n), \tag{2.5a}$$

where the *coset representatives* $[1]$, $[2]$, and $[3]$ are defined in this case such that

$$[1] = \begin{pmatrix} 1/2 \\ \vdots \\ 1/2 \\ 1/2 \end{pmatrix}, \quad [2] = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \quad [3] = \begin{pmatrix} 1/2 \\ \vdots \\ 1/2 \\ -1/2 \end{pmatrix}.$$

The D_n^* lattice may also be constructed via integer linear combination of the columns of the $n \times n$ basis matrix

$$B_{D_n^*} = \begin{pmatrix} 1 & & & 0 & 0.5 \\ & 1 & & & 0.5 \\ & & \ddots & & \vdots \\ & & & 1 & 0.5 \\ 0 & & & & 0.5 \end{pmatrix}. \tag{2.5b}$$

It is important to recognize that, for $n \geq 5$, the contact graph of the D_n^* lattice is simply two disjoint nets given by the contact graphs of the $\mathbb{Z}^n$ and shifted $\mathbb{Z}^n$ sets of lattice points upon which D_n^* may be built [see (2.5a)]. Thus, as suggested by Conway

& Sloane (1997), we introduce, for $n \geq 4$, a *generalized net* formed by connecting each node of the unshifted $\mathbb{Z}^n$ set to the 2^n nearest nodes on the shifted $\mathbb{Z}^n$ set, and each node on the shifted $\mathbb{Z}^n$ set to the 2^n nearest nodes on the unshifted $\mathbb{Z}^n$ set. The resulting net, of coordination number 2^n, is uninodal, but is *not* a contact graph of the corresponding sphere packing.

The packing D_n^+, reasonably identified as the *offset checkerboard packing*, is an n-dimensional extension of the 3-dimensional *diamond* packing, and is defined simply as

$$D_n^+ = D_n \cup ([1] + D_n); \tag{2.6}$$

note that D_n^+ is a lattice packing only for even n, and that D_3^+ is the *diamond packing* (for further discussion, see §2.4.4).

2.3.4 The zero-sum lattice A_n, its dual A_n^*, and the glued zero-sum lattices A_n^r

The *zero-sum lattice*, A_n, may be thought of as an n-dimensional extension of the 2-dimensional *triangular lattice*; in 3 dimensions, $A_3 \cong D_3$. It is defined

$$A_n = \left\{(x_0, \ldots, x_n) \in \mathbb{Z}^{n+1} : x_0 + \ldots + x_n = 0\right\}, \tag{2.7a}$$

and may be constructed via integer linear combination of the columns of the $(n+1) \times n$ basis matrix

$$B_{A_n} = \begin{pmatrix} -1 & & & 0 \\ 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \\ 0 & & & 1 \end{pmatrix}, \qquad \text{with} \qquad \mathbf{n}_{A_n} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}. \tag{2.7b}$$

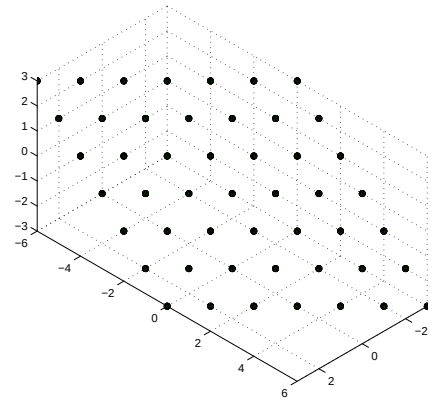

Figure 2.6: A cloud of points on the A_2 lattice, defined on a plane in $\mathbb{R}^3$. Note that the normal vector $\mathbf{n}_{A_2} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^T$ points directly out of the page in this view.

Notice that A_n is constructed here via n basis vectors in $n+1$ dimensions. The resulting lattice lies in an n-dimensional subspace in $\mathbb{R}^{n+1}$; this subspace is normal to the vector $\mathbf{n}_{A_n}$. An illustrative example is A_2, the triangular $2D$ lattice, which may conveniently be constructed on a plane in $\mathbb{R}^3$ (see Figure 2.6).

Note that, starting from a (2D) triangular configuration of oranges or cannonballs (see Figure 2.3a), one can stack additional layers of oranges in a trangular configuration on top, appropriately offset from the base layer, to build up the (3D) FCC configuration mentioned previously (see Figure 2.3a). This idea is referred to as lamination, and will be extended further in §2.3.6 when considering the Λ_n family of lattices.

Also note that, in the special case of $n = 2$, the A_2 lattice may also be written as

$$A_2 \cong R_2 \cup (\mathbf{a} + R_2), \qquad \text{where} \quad \mathbf{a} = \begin{pmatrix} 1/2 \\ \sqrt{3}/2 \end{pmatrix} \tag{2.7c}$$

and R_2 is the *rectangular grid* (not a lattice, nor even a nonlattice packing) obtained by stretching the $\mathbb{Z}^2$ lattice in the second element by a factor of $\sqrt{3}$.

The dual of the zero-sum lattice, denoted A_n^*, may be written as

$$A_n^* = \bigcup_{s=0}^{n} ([s] + A_n), \tag{2.8a}$$

where the $n+1$ coset representatives $[s]$, for $s = 0, \ldots, n$, are defined such that the k'th component of the vector $[s]$ is

$$[s]_k = \begin{cases} \frac{s}{n+1} & k \leq n+1-s, \\ \frac{s-n-1}{n+1} & \text{otherwise.} \end{cases} \tag{2.8b}$$

The A_n^* lattice may be constructed via integer linear combination of the columns of the $(n+1) \times n$ basis matrix

$$B_{A_n^*} = \begin{pmatrix} 1 & 1 & \cdots & 1 & \frac{-n}{n+1} \\ -1 & & & 0 & \frac{1}{n+1} \\ & -1 & & & \frac{1}{n+1} \\ & & \ddots & & \vdots \\ & & & -1 & \frac{1}{n+1} \\ 0 & & & & \frac{1}{n+1} \end{pmatrix}, \qquad \text{with} \qquad \mathbf{n}_{A_n^*} = \mathbf{n}_{A_n}. \tag{2.8c}$$

A related family of lattice packings, developed in §12 of Coxeter (1951) and reasonably identified as the *glued zero-sum lattices* A_n^r, is a family of lattices somewhere between A_n and A_n^* [as given in (2.8a)] defined via the union of r translates of A_n for $n \geq 5$:

$$A_n^r = A_n \cup ([s] + A_n) \cup ([2s] + A_n) \cup \ldots \cup ([(r-1)s] + A_n), \quad \text{where} \quad r \cdot s = n+1, \tag{2.9}$$

where the components of the "glue" vectors $[s]$ are specified in (2.8b), and where r and s are integer divisors of $(n+1)$ with $1 < s < n+1$ and $1 < r < n+1$, excluding the case $\{r = 2, s = 3\}$ for $n = 5$. The lattices A_9^5, A_{11}^4, A_{13}^7, A_{14}^5, A_{15}^8, A_{17}^9, A_{19}^{10}, A_{20}^7, and A_{21}^{11} are found to have especially good covering thickness, with the last four currently the thinnest coverings available in their respective dimensions (see Baranovskii 1994, Anzin 2002, and Sikirić, Schürmann, & Vallentin 2008). Note also that $A_7^2 \cong E_7$, $A_7^4 \cong E_7^*$, and $A_8^3 \cong E_8$, each of which is discussed further below.

Note finally that the contact graphs of some of the A_n^r lattices, such as A_9^5 and A_{11}^4, are disjoint nets given by the contact graphs of the A_n and shifted A_n sets of lattice points upon which these glued zero-sum lattices are built [see (2.9)]. Thus, as in the case of D_n^* for $n > 4$ as discussed in §2.3.3, a *generalized net* may be formed by connecting

each node of the unshifted A_n set to the nearest nodes on the shifted A_n set. Again, the resulting net is uninodal, but is not a contact graph of the corresponding sphere packing.

2.3.5 The Gosset lattice $E_8 \cong E_8^*$, E_7, E_7^*, E_6, and E_6^*

The *Gosset lattice* $E_8 \cong E_8^*$, which has a (remarkable) kissing number of $\tau = 240$, may be defined simply as

$$E_8 = D_8^+, \tag{2.10a}$$

and may be constructed via integer linear combination of the columns of the 8×8 basis matrix

$$B_{E_8} = \begin{pmatrix} 2 & -1 & & & & & 0 & 1/2 \\ & 1 & -1 & & & & & 1/2 \\ & & 1 & -1 & & & & 1/2 \\ & & & 1 & -1 & & & 1/2 \\ & & & & 1 & -1 & & -1/2 \\ & & & & & 1 & -1 & -1/2 \\ & & & & & & 1 & -1/2 \\ 0 & & & & & & & -1/2 \end{pmatrix}. \tag{2.10b}$$

The lattice E_7 is defined by restricting E_8, as constructed above, to a 7-dimensional subspace,

$$E_7 = \{(x_1, \ldots, x_8) \in E_8 : x_1 + \ldots + x_8 = 0\}, \tag{2.11a}$$

and may be constructed directly via integer linear combination of the columns of the 8×7 basis matrix

$$B_{E_7} = \begin{pmatrix} -1 & & & & & 0 & 1/2 \\ 1 & -1 & & & & & 1/2 \\ & 1 & -1 & & & & 1/2 \\ & & 1 & -1 & & & 1/2 \\ & & & 1 & -1 & & -1/2 \\ & & & & 1 & -1 & -1/2 \\ & & & & & 1 & -1/2 \\ 0 & & & & & & -1/2 \end{pmatrix}, \qquad \text{with} \qquad \mathbf{n}_{E_7} = \begin{pmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{pmatrix}. \tag{2.11b}$$

The dual of the E_7 lattice may be written as

$$E_7^* = E_7 \cup ([1] + E_7), \qquad \text{where} \quad [1] = \begin{pmatrix} 1/4 \\ \vdots \\ 1/4 \\ -3/4 \\ -3/4 \end{pmatrix}, \tag{2.12a}$$

and may be constructed directly via integer linear combination of the columns of the 8×7 basis matrix

$$B_{E_7^*} = \begin{pmatrix} -1 & & & & & 0 & -3/4 \\ 1 & -1 & & & & & -3/4 \\ & 1 & -1 & & & & 1/4 \\ & & 1 & -1 & & & 1/4 \\ & & & 1 & -1 & & 1/4 \\ & & & & 1 & -1 & 1/4 \\ & & & & & 1 & 1/4 \\ 0 & & & & & & 1/4 \end{pmatrix}, \qquad \text{with} \qquad \mathbf{n}_{E_7^*} = \mathbf{n}_{E_7}. \tag{2.12b}$$

The lattice E_6 is defined by further restricting E_7, as defined in (2.11), to a 6-dimensional subspace,

$$E_6 = \{(x_1, \ldots, x_8) \in E_7 : x_1 + x_8 = 0\}, \tag{2.13a}$$

and may be constructed directly via integer linear combination of the columns of the 8×6 basis matrix

$$B_{E_6} = \begin{pmatrix} & & & & 0 & 1/2 \\ -1 & & & & & 1/2 \\ 1 & -1 & & & & 1/2 \\ & 1 & -1 & & & 1/2 \\ & & 1 & -1 & & -1/2 \\ & & & 1 & -1 & -1/2 \\ & & & & 1 & -1/2 \\ 0 & & & & & -1/2 \end{pmatrix}, \qquad \text{with} \qquad N_E = \begin{pmatrix} 1 & 1/2 \\ 0 & 1/2 \\ 0 & 1/2 \\ 0 & 1/2 \\ 0 & 1/2 \\ 0 & 1/2 \\ 0 & 1/2 \\ 1 & 1/2 \end{pmatrix} = \begin{pmatrix} | & | \\ \mathbf{n}_{E_6} & \mathbf{n}_{E_7} \\ | & | \end{pmatrix}. \tag{2.13b}$$

The dual of the E_6 lattice may be written as

$$E_6^* = E_6 \cup ([1] + E_6) \cup ([2] + E_6), \qquad \text{where} \quad [1] = \begin{pmatrix} 0 \\ -2/3 \\ -2/3 \\ 1/4 \\ \vdots \\ 1/4 \\ 0 \end{pmatrix}, \quad [2] = -[1], \tag{2.14a}$$

and may be constructed directly via integer linear combination of the columns of the 8×6 basis matrix

$$B_{E_6^*} = \begin{pmatrix} & & & 0 & 0 & 1/2 \\ -1 & & & & 2/3 & 1/2 \\ 1 & -1 & & & 2/3 & 1/2 \\ & 1 & -1 & & -1/3 & 1/2 \\ & & 1 & -1 & -1/3 & -1/2 \\ & & & 1 & -1/3 & -1/2 \\ & & & & -1/3 & -1/2 \\ 0 & & & & 0 & -1/2 \end{pmatrix}, \qquad \text{with} \qquad N_{E^*} = N_E. \tag{2.14b}$$

2.3.6 The laminated lattices Λ_n and K_n lattices

The lattices in the Λ_n and K_n families can be built up one dimension, or "laminate", at a time, starting from the integer lattice ($\mathbb{Z} \cong \Lambda_1 \cong K_1$), to triangular ($A_2 \cong \Lambda_2 \cong K_2$), to FCC ($A_3 \cong D_3 \cong \Lambda_3 \cong K_3$), all the way up (one layer at a time) to the remarkable Leech lattice ($\Lambda_{24} \cong K_{24}$). Both families of lattices may in fact be extended (but not uniquely) to at least $n = 48$.

The Leech lattice, Λ_{24}, is the unique lattice in $n = 24$ dimensions with a (remarkable) kissing number of $\tau = 196,560$. It may be constructed via integer linear combination of the columns of the 24×24 basis matrix $B_{\Lambda_{24}}$, which is depicted here in the celebrated Miracle Octad Generator (MOG) coordinates (see Curtis 1976 and Conway & Sloane 1999):

$$B_{\Lambda_{24}} = \frac{1}{\sqrt{8}} \begin{pmatrix}
8 & 4 & 4 & 4 & 4 & 4 & 4 & 2 & 4 & 4 & 4 & 2 & 4 & 2 & 2 & 2 & 4 & 2 & 2 & 2 & & & -3 \\
 & 4 & & & & & & 2 & & & & 2 & & 2 & & & & & & 2 & 2 & & 1 \\
 & & 4 & & & & & 2 & & & & 2 & & & 2 & & & 2 & & & 2 & & 1 \\
 & & & 4 & & & & 2 & & & & 2 & & & & 2 & & & 2 & & 2 & & 1 \\
 & & & & 4 & & & 2 & & & & & & 2 & 2 & 2 & & 2 & 2 & 2 & 2 & & 1 \\
 & & & & & 4 & & 2 & & & & & & 2 & & & & & 2 & & & & 1 \\
 & & & & & & 4 & 2 & & & & & & & 2 & & & & & 2 & & & 1 \\
 & & & & & & & 2 & & & & & & & & 2 & & 2 & & & & & 1 \\
 & & & & & & & & 4 & & & 2 & & 2 & 2 & 2 & & 2 & 2 & 2 & 2 & 2 & 2 & 1 \\
 & & & & & & & & & 4 & & 2 & & 2 & & & & 2 & & & 2 & & 1 \\
 & & & & & & & & & & 4 & 2 & & & 2 & & & & 2 & & & 2 & 1 \\
 & & & & & & & & & & & 2 & & & & 2 & & & & 2 & & & 1 \\
 & & & & & & & & & & & & 4 & 2 & 2 & 2 & & & & 2 & 2 & 2 & 1 \\
 & & & & & & & & & & & & & 2 & & & & & & & 2 & & 1 \\
 & & & & & & & & & & & & & & 2 & & & & & & & 2 & 1 \\
 & & & & & & & & & & & & & & & 2 & & & & & & & 1 \\
 & & & & & & & & & & & & & & & & 4 & 2 & 2 & 2 & 2 & 2 & 2 & 1 \\
 & & & & & & & & & & & & & & & & & 2 & & & 2 & & 1 \\
 & & & & & & & & & & & & & & & & & & 2 & & & 2 & 1 \\
 & & & & & & & & & & & & & & & & & & & 2 & & & 1 \\
 & & & & & & & & & & & & & & & & & & & & 2 & 2 & 2 & 1 \\
 & & & & & & & & & & & & & & & & & & & & & 2 & & 1 \\
 & & & & & & & & & & & & & & & & & & & & & & 2 & 1 \\
 & & & & & & & & & & & & & & & & & & & & & & & 1
\end{pmatrix}.$$

As in the $E_8 \to E_7 \to E_6$ progression described in §2.3.5, the Λ_n lattices for $n = 23, 22, \ldots, 1$ may all be constructed by restricting the Λ_{24} lattice to smaller and smaller subspaces via the normal vectors assembled in the matrix[12]

[12]There are, of course, *many* equivalent constructions of Λ_1 through Λ_{23} via restriction of Λ_{24}, and the available literature on the subject considers these symmetries at length. The convenient form of N_Λ depicted here was deduced, with some effort, from Figure 6.2 of Conway & Sloane (1999).

$$
N_\Lambda = \begin{pmatrix}
 & & 1 & & & & & & & & & & & & & & & & & & & & & \\
 & & & 1 & & & & & & & & & & & & & & & & & & & & \\
 & 1 & 1 & & & & & & & & & & & & & & & & & & & & & \\
1 & 1 & 1 & & & & & & & & & & & & & & & & & & & & & \\
 & & & & 1 & & & & & & & & & & & & & & & & & & & \\
 & & & & & 1 & -1 & & & & & & & & & & & & & & & & & \\
 & & & & & & 1 & -1 & & & & & & & & & & & & & & & & \\
 & & & & & & & 1 & & & & & & & & & & & & & & & & \\
 & & & & & & & & 1 & & & & & & & & & & & & & & & \\
 & & & & & & & & & 1 & -1 & & & & & & & & & & & & & \\
 & & & & & & & & & & 1 & -1 & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & 1 & & & & & & & & & \\
 & & & & & & & & & & & & & 1 & & 1 & & & & & & & & \\
 & & & & & & & & & & & & 1 & & & 1 & & & & & & & & \\
 & & & & & & & & & & & 1 & & & & & & & & & & & & \\
 & & & & & & & & & & 1 & & & & & 1 & & & & & & & & \\
 & & & & & & & & & & & & & & & & & 1 & & & & & & \\
 & & & & & & & & & & & & & & & & 1 & 1 & & & & & & \\
 & & & & & & & & & & & & & & & 1 & & 1 & & & & & & \\
 & & & & & & & & & & & & & & 1 & & & 1 & & & & & & \\
 & & & & & & & & & & & & & & & & & & 1 & & & & & \\
 & & & & & & & & & & & & & & & & & & & 1 & -1 & & & \\
 & & & & & & & & & & & & & & & & & & & & 1 & -1 & & \\
 & & & & & & & & & & & & & & & & & & & & & & 1 &
\end{pmatrix}
$$

$$
= \begin{pmatrix} \mathbf{n}_{\Lambda_0} & \dots & \mathbf{n}_{\Lambda_{23}} \end{pmatrix}.
$$

Thus, the Λ_{23} lattice is obtained from the points of the Λ_{24} lattice in $\mathbb{R}^{24}$(which themselves are generated via integer linear combination of the columns of $B_{\Lambda_{24}}$) which lie in the 23-dimensional subspace orthogonal to $\mathbf{n}_{\Lambda_{23}}$. Similarly, the Λ_{22} lattice is obtained from the points of the Λ_{24} lattice which lie in the 22-dimensional subspace orthogonal to both $\mathbf{n}_{\Lambda_{23}}$ and $\mathbf{n}_{\Lambda_{22}}$, etc. Noting the block diagonal structure of N_Λ, it follows

that Λ_n may be constructed using the basis matrix, denoted B_{Λ_n}, given by the $n \times n$ submatrix in the upper-left corner of $B_{\Lambda_{24}}$ for any $n \in \mathsf{N}_1 = \{21, 20, 16, 9, 8, 5, 4\}$. For the remaining dimensions, $n \in \mathsf{N}_2 = \{19, 18, 17, 15, 14, 13, 12, 11, 10, 7, 6, 3, 2, 1\}$, Λ_n may be constructed via the appropriate restriction of the lattice generated by the next larger basis matrix in the set N_1; for example, Λ_{14} may be constructed in $\mathbb{R}^{16}$ via restriction of the lattice generated by the basis matrix $B_{\Lambda_{16}}$ to the subspace normal to the vectors (in $\mathbb{R}^{16}$) given by the first 16 elements of $\mathbf{n}_{\Lambda_{15}}$ and $\mathbf{n}_{\Lambda_{14}}$.

A similar sequence of lattices, denoted K_n, may be constructed via restriction of the Leech lattice (generated via $B_{\Lambda_{24}}$) in a similar fashion (for details, see Figure 6.3 of Conway & Sloane 1999). Lattices from the Λ_n and/or K_n families have the maximal packing densities and kissing numbers amongst all lattices for the entire range considered here, $1 \leq n \leq 24$. Note that the Λ_n and K_n families are not equivalent in the range $7 \leq n \leq 17$, with Λ_n being superior to K_n by all four metrics introduced in §2.2 at most values of n in this range, except for the narrow range $11 \leq n \leq 13$, where in fact K_n has a slight advantage. Note also that there is some flexibility in the definition of the lattices Λ_{11}, Λ_{12}, and Λ_{13}; the branch of the Λ_n family considered here is that which maximizes the kissing number τ in this range of n, and thus the corresponding lattices are denoted $\Lambda_{11}^{\max}$, $\Lambda_{12}^{\max}$, and $\Lambda_{13}^{\max}$. Note that K_{12} is referred to as the Coxeter-Todd lattice and Λ_{16} is referred to as the Barnes-Wall lattice.

2.3.7 Some numerically-generated lattices for thin coverings in dimensions 6-15

Recall from §2.3.1 that an n-dimensional real lattice may be defined as an integer linear combination of a set of n basis vectors $\mathbf{b}^i$ defined in $\mathbb{R}^{n+m}$ for $m \geq 0$; that is, any lattice point may be written as

$$\mathbf{x} = \xi_1 \mathbf{b}^1 + \xi_2 \mathbf{b}^2 + \ldots + \xi_n \mathbf{b}^n = B\boldsymbol{\xi},$$

where the elements $\{\xi_1, \ldots, \xi_n\}$ of the vector $\boldsymbol{\xi}$ are taken as integers. The square of the distance of any lattice point from the origin is thus given by $f(\boldsymbol{\xi}) = \boldsymbol{\xi}^T A \boldsymbol{\xi}$, where $A \triangleq B^T B$ is known as the *Gram matrix* associated with the lattice in question, and the function $f(\boldsymbol{\xi})$ is referred to as the corresponding *quadratic form* [note that each term of $f(\boldsymbol{\xi})$ is quadratic in the elements of $\boldsymbol{\xi}$]. All of the lattices studied thus far, when scaled appropriately, are characterized by Gram matrices with *integer elements*, and thus their corresponding quadratic forms $f(\boldsymbol{\xi})$ have integer coefficients (and are thus referred to as *integral quadratic forms*).

There is particular mathematical interest in discovering (or generating numerically) both lattice and non-lattice packings which minimize covering thickness and/or packing density. The numerical approach to this problem studied in Schürmann & Vallentin (2006) and Sikirić, Schürmann, & Vallentin (2008) has generated new lattices in dimensions 6-15 with the thinnest covering thicknesses known amongst all lattices. The lattice so generated in dimension 7 happens to correspond to an integral quadratic form, but the others, apparently, do not. Gram matrices A corresponding to these 10 lattices (denoted L_6^{c1}, L_7^c, L_8^c, …, L_{15}^c) are available at *http://fma2.math.uni-magdeburg.de/~latgeo/covering_table.html*;
(nonunique) basis matrices B corresponding to each of these lattices may be generated simply by taking the Cholesky decomposition of the corresponding Gram matrix, as $A = B^T B$.

2.3.8 Discussion

For all of the dense lattices described thus far, as well as for the rare packings and nets described in §2.4, Tables 2.2 and 2.3 list the known values of the packing density

Δ, the covering thickness Θ, and the mean squared quantization error per dimension, G. Table 2.2 also lists the coordination sequence through $k = 10$ of the corresponding net, as well as its local topological density td_{10}. If this net is a contact graph, the coordination number (that is, the first element of the coordination sequence) is equal to the kissing number of the corresponding packing; if this net is *not* a contact graph, it is marked with a **G**, and the kissing number τ of the corresponding sphere packing is listed in parentheses.

The other information appearing in Table 2.2 is described further in §2.4. Note that Table 2.2 alone has 8 columns and over 100 rows, with those results which we believe to be new denoted in italics. The original source of each of the several hundred existing results reported can not feasibly be spelled out here. Suffice it to say that the vast majority of those existing results related to lattices are discussed in Conway & Sloane (1999) and in the On-Line Encyclopedia of Integer Sequences (on the web at `http://www.research.att.com/~njas/sequences/`), where a large number of the original references are listed in detail. The vast majority of those existing results related to 3D nets (see §2.4), *including clear drawings of each* as well as detailed lists of original references, are given in the Reticular Chemistry Structure Resource, available online at, e.g., `http://rcsr.anu.edu.au/nets/fcu`, where "`fcu`" may be replaced by any of the lowercase boldface three-letter identifiers given in Table 2.2 and §2.4; for further discussion of this database and others, see O'Keeffe et al. (2008), Treacy et al. (2004), Blatov (2006), and Hyde et al. (2006). Note also that there are hundreds of new results reported in Tables 2 and 3, as denoted in italics; most of these are the result of painstaking numerical simulation, some of which tooks weeks of CPU time (on a quad-core 3GHz Intel Xeon server) to complete.

Note finally that there are a variety of simple ways to quantize to the nearest lattice point; for an introduction, see Appendix A.

Table 2.2: Known characteristics of selected lattices in dimension $1 \leq n \leq 9$. Boldface denotes values known, or believed, to be optimal among all lattices at that dimension n. Note that $\mathbb{Z}^n$ is referred to as the cubic lattice, K_{12} is referred to as the Coxeter-Todd lattice, Λ_{16} is referred to as the Barnes-Wall lattice, and Λ_{24} is referred to as the Leech lattice. The symbol $\approx$ denotes approximate values estimated via Monte Carlo integration; all other results reported have been determined in the literature analytically (see Conway & Sloane 1998). The symbol $\leq$ denotes a bound, not an exact value. Blank entries appear to be unavailable in the published literature on this subject.

n	packing	**net**	Δ	Θ	G	**coordination sequence**	td_{10}	**point symbol vertex symbol**
1	$\mathbb{Z}, \Lambda_1$	integer	**1**	**1**	**0.083333**	**2**, 2, 2, 2, 2, 2, 2, 2, 2, 2	**21**	$*$
2	A_2, A_2^*, Λ_2	triangular	**0.90690**	**1.2092**	**0.080188**	**6**, 12, 18, 24, 30, 36, 42, 48, 54, 60	**331**	$3^6.4^6.5^3$
	$\mathbb{Z}^2, D_2, D_2^*, D_2^+$	square	0.78540	1.5708	0.083333	4, 8, 12, 16, 20, 24, 28, 32, 36, 40	221	$4.4.4.4.*.*$
	$A_2^+, {}^\intercal A_2^*$	honeycomb	0.60460	2.4184	*0.09623*	3, 6, 9, 12, 15, 18, 21, 24, 27, 30	166	6.6.6
	$\hat{A}_2^+, {}^\intercal \hat{A}_2^*$	augmented honeycomb	**0.39067**	*5.832*	*0.1652*	3, 4, 6, 8, 12, 14, 15, 18, 21, 22	124	3.12.12

Table 2.2 continued

n	packing	**net**	Δ	Θ	G	**coordination sequence**	td_{10}	**point symbol vertex symbol**
3	D_3, A_3, Λ_3	**fcu**	**0.74048**	2.0944	0.078745	**12**, 42, 92, 162, 252, 362, 492, 642, 812, 1002	3871	$3^{24}.4^{36}.5^6$
		hcp	**0.74048**	2.0944	0.078745	**12**, 44, 96, 170, 264, 380, 516, 674, 852, 1052	**4061**	$3^{24}.4^{33}.5^9$
	D_3^*, A_3^*	**bcu**	0.68017	**1.4635**	**0.078543**	8, 26, 56, 98, 152, 218, 296, 386, 488, 602	2331	$4^{24}.6^4$
	$\mathbb{Z}^3$	**pcu**	0.52360	2.7207	0.083333	6, 18, 38, 66, 102, 146, 198, 258, 326, 402	1561	$4^{12}.6^3$
		qtz, V_3^{60}	0.39270	*2.0405*	*0.08534*	4, 12, 30, 52, 80, 116, 156, 204, 258, 318	1231	$6.6.6_2.6_2.8_7.8_7$
	A_3^+, D_3^+	**dia**, V_3^{90}	0.34009	2.7207	*0.09114*	4, 12, 24, 42, 64, 92, 124, 162, 204, 252	981	$6_2.6_2.6_2.6_2.6_2.6_2$

Table 2.2 continued

n	packing	**net**	Δ	Θ	G	**coordination sequence**	td_{10}	**point symbol vertex symbol**
3		**lon**	0.34009	*3.3068*	*0.09139*	4, 12, 25, 44, 67, 96, 130, 170, 214, 264	1027	$6_2.6_2.6_2.6_2.6_2.6_2$
	${}^{\mathsf{T}}A_3^*$	**sod**	0.2777	*8.781*	*0.1092*	4, 10, 20, 34, 52, 74, 100, 130, 164, 202	791	4.4.6.6.6.6
	$\hat{A}_3^+$	**dia-a**	0.12354	*9.1723*	*0.1511*	4, 6, 12, 18, 36, 48, 60, 78, 108, 126	497	$3.12_2.3.12_2.3.12_2$
	${}^{\mathsf{T}}\hat{A}_3^*$	**sod-a**	**0.1033**	*28.26*	*0.1943*	4, 6, 12, 17, 28, 38, 52, 64, 84, 104	410	3.8.3.12.3.12
		qzd, T_3^{60}	0.6046	*2.1549*	*0.08151*	**G**: 4, 12, 36, 72, 122, 188, 264, 354, 456, 570 $(\tau = 8)$	**2079**	$7_2.*.7_3.7_3.7_3.7_3$
		cds, T_3^{90}	0.52360	2.7207	0.08333	**G**: 4, 12, 30, 58, 94, 138, 190, 250, 318, 394 $(\tau = 6)$	1489	$6.6.6.6.6_2.*$
		nbo, S_3	0.39270	*3.1416*	*0.08602*	4, 12, 28, 50, 76, 110, 148, 194, 244, 302	1169	$6_2.6_2.6_2.6_2.8_2.8_2$
		bto $(\alpha = 60^\circ)$,	0.2687	*3.0042*	*0.09129*	3, 6, 12, 24, 43, 64, 91, 124, 160, 202	730	$10.10_2.10_2$
		Y_3^{60} $(\alpha \approx 70.5^\circ)$	*0.2551*	*2.7251*	*0.09217*			
		ths $(\alpha = 60^\circ)$,	0.2327	*4.3099*	*0.09706*	3, 6, 12, 24, 38, 56, 77, 102, 129, 160	608	$10_2.10_4.10_4$
		Y_3^{90} $(\alpha \approx 70.5^\circ)$	*0.2207*	*3.518*	*0.09817*			

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
4	**srs**	0.1851	*3.4281*	*0.1072*	3, 6, 12, 24, 35, 48, 69, 86, 108, 138	530	$10_5.10_5.10_5$
	srs-a	**0.0555**	*9.739*	*0.1882*	3, 4, 6, 8, 12, 16, 24, 32, 48, 54	208	$3.20_5.20_5$
	D_4, D_4^*, Λ_4	**0.61685**	2.4674	**0.076603**	**24**, 144, 456, 1056, 2040, 3504, 5544, 8256, 11736, 16080	**48,841**	$3^{96}.4^{168}.5^{12}$
					G: 16, 80, 240, 544, 1040, 1776, 2800, 4160, 5904, 8080 $(\tau = 24)$	24,641	$4^{112}.6^8$
	A_4	0.55173	3.1780	0.078020	20, 110, 340, 780, 1500, 2570, 4060, 6040, 8580, 11750	35,751	$3^{60}.4^{120}.5^{10}$
	A_4^*	0.44138	**1.7655**	0.077559	10, 50, 150, 340, 650, 1110, 1750, 2600, 3690, 5050	15,401	$4^{40}.6^5$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
4	$\mathbb{Z}^4, D_4^+$	0.30843	4.9348	0.08333	8, 32, 88, 192, 360, 608, 952, 1408, 1992, 2720	8361	$4^{24}.6^4$
	A_4^+	0.17655	*6.3558*	*0.08827*	5, 20, 50, 110, 200, 340, 525, 780, 1095, 1500	4626	6^{10}
	${}^{\mathsf{T}}A_4^*$	0.10593	*42.4*	*0.1221*	5, 15, 35, 70, 125, 205, 315, 460, 645, 875	2751	$4^5.6^5$
	$\hat{A}_4^+$	0.03354	*23.82*	*0.1398*	*5, 8, 20, 32, 80, 116, 170, 236, 380, 482*	*1530*	$3^6.12^4$
	T_4^{90}	*0.3084*	*4.935*	*0.08333*	**G**:4,12,36,92,200,384, 664,1056,1576,2240 $(\tau = 8)$	*6265*	$8_3.8_3.8_3.8_3.8_4.*$
	S_4	*0.1542*	*3.855*	*0.08692*	*4, 12, 36, 84, 172, 292, 468, 692, 988, 1348*	*4097*	$8_2.8_2.8_5.8_5.8_5.8_5$
	V_4^{90}	*0.1187*	*5.814*	*0.09333*	*4, 12, 36, 74, 136, 228, 352, 518, 732, 994*	*3087*	$8_6.8_6.8_7.8_7.8_7.8_7$
	Y_4^{90}	*0.06793*	*6.458*	*0.09736*	*3, 6, 12, 24, 48, 90, 146, 230, 336, 478*	*1374*	$12_2.12_2.12_2$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
5	D_5, Λ_5	**0.46526**	4.5977	0.075786	**40**, 370, 1640, 4930, 11752, 24050, 44200, 75010, 119720, 182002	**463,715**	$3^{240}.4^{520}.5^{20}$
	A_5	0.37988	5.9218	0.077647	30, 240, 1010, 2970, 7002, 14240, 26070, 44130, 70310, 106752	272,755	$3^{120}.4^{300}.5^{15}$
	D_5^*	0.32899	2.4982	**0.075625**	**G**: 32, 242, 992, 2882, 6752, 13682, 24992, 42242, 67232, 102002 $(\tau = 10)$	261,051	$4^{480}.6^{16}$
	D_5^+	0.28736	5.2638	*0.07784*	*16, 120, 480, 1410, 3296, 6712, 12256, 20770, 33056, 50232*	*128,349*	$4^{80}.6^{40}$
	A_5^*	0.25543	**2.1243**	0.076922	12, 72, 272, 762, 1752, 3512, 6372, 10722, 17012, 25752	66,241	$4^{60}.6^{6}$
	$\mathbb{Z}^5$	0.16449	9.1955	0.083333	10, 50, 170, 450, 1002, 1970, 3530, 5890, 9290, 14002	36,365	$4^{40}.6^{5}$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
5	A_5^+	0.08514	*8.8223*	*0.08646*	6, 30, 90, 240, 510, 1010, 1770, 2970, 4626, 7002	18,255	6^{15}
	${}^{\mathsf{T}}A_5^*$	0.035174	*254.9*	*0.1349*	6, 21, 56, 126, 252, 461, 786, 1266, 1946, 2877	7798	$4^9.6^6$
	$\hat{A}_5^+$	0.008055	*35.81*	*0.1313*	*6, 10, 30, 50, 150, 230, 390, 570, 1050, 1420*	*3907*	$3^{10}.12^5$
	T_5^{90}	*0.16449*	*9.1955*	*0.08333*	**G***: 4, 12, 36, 100, 258, 610, ?* $(\tau = 10)$	*?*	$8_2.8_2.8_2.8_2.10_6.*$
	S_5	*0.05140*	*9.310*	*0.08666*	*4, 12, 36, 100, 244, 514, 980, 1682, 2724, 4162*	*10,459*	$8.8.8.8.8_2.8_2$
	V_5^{60}	*0.04786*	*8.4884*	*0.08753*	*4, 12, 36, 100, 248, 522, 988, 1724, 2800, 4324*	*10,759*	$8.8.8.8.8_2.8_2$
	Y_5^{60}	*0.03516*	*254.8*	*0.1350*	*3, 6, 12, 24, 48, 90, 168, 312, 556, 914*	*2134*	$12_2.12_2.12_2$
	T_5^{60}	*0.02478*	*6.2578*	*0.09038*	**G***: 4, 12, 36, 100, 268, ?* $(\tau = 14)$	*?*	$8_2.8_2.8_2.8_2.11_{10}.*$
	V_5^{90}	*0.02478*	*6.016*	*0.09037*	*4, 12, 36, 100, 220, 428, 752, 1254, 1944, 2924*	*7675*	$8.8.8.8.8_2.8_2$
	Y_5^{90}	*0.01858*	*11.19*	*0.09605*	*3, 6, 12, 24, 48, 90, 168, 312, 532, 872*	*2068*	$12_2.12_2.12_2$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
6	E_6, Λ_6	**0.37295**	7.0722	0.074347	**72**, 1062, 6696, 26316, 77688, 189810, 405720, 785304, 1408104	2,900,773	$3^{720}.4^{1800}.5^{36}$
	E_6^*	0.33151	2.6521	**0.074244**	54, 828, 5202, 20376, 60030, 146484, 312858, 605232, 1084806, 1830060	**4,065,931**	$3^{270}.4^{1134}.5^{27}$
	D_6	0.32298	8.7205	0.075591	60, 792, 4724, 18096, 52716, 127816, 271908, 524640, 938652, 1581432	3,520,837	$3^{480}.4^{1260}.5^{30}$
	D_6^+	0.27252	5.1677	*0.07459*	*32, 332, 1824, 6776, 19488, 46980, 99680, 192112, 343584, 578876*	*1,289,685*	$4^{480}.6^{16}$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
6	A_6	0.24415	9.8401	0.077466	42, 462, 2562, 9492, 27174, 65226, 137886, 264936, 472626, 794598	1,775,005	$3^{210}.4^{630}.5^{21}$
	D_6^*	0.16149	4.3603	0.075120	**G**: 64, 728, 4032, 14896, 42560, 102024, 215488, 413792, 737856, 1240120 ($\tau = 12$)	244,069	$4^{1984}.6^{32}$
	A_6^*	0.13453	2.5511	0.076490	14, 98, 462, 1596, 4410, 10374, 21658, 41272, 73206, 122570	275,661	$4^{84}.6^{7}$
	L_6^{c1}	*0.31853*	**2.4648**	?	*32*, ?	?	?
	$\mathbb{Z}^6$	0.08075	17.441	0.08333	12, 72, 292, 912, 2364, 5336, 10836, 20256, 35436, 58728	134,245	$4^{60}.6^{6}$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
6	A_6^+	0.03844	*19.681*	*0.08525*	7, 42, 147, 462, 1127, 2562, 5047, 9492, 16317, 27174	62,378	6^{21}
	${}^{\mathsf{T}}A_6^*$	0.010459	*1836.5*	*0.14712*	7, 28, 84, 210, 462, 924, 1715, 2996, 4977, 7924	19,328	$4^{14}.6^7$
	$\hat{A}_6^+$	0.001774	*99.91*	*0.1259*	*7, 12, 42, 72, 252, 402, 777, 1182, 2457, 3492*	*6,496*	$3^{15}.12^6$
	T_6^{90}	*0.08075*	*17.441*	*0.08333*	**G***: 4, 12, 36, 100, ?* ($\tau = 12$)	*?*	?
	S_6	*0.01514*	*9.78*	*0.08601*	*4, 12, 36, 100, 276, 660, 1484, 2920, ?*	*?*	$8.8.8.8.8_2.8_2$
	V_6^{90}	*9.740e-3*	*19.79*	*0.09322*	*4, 12, 36, 100, 276, 610, 1284, 2346, 4152, 6792*	*15,613*	$8.8.8.8.8_2.8_2$
	Y_6^{90}	*4.640e-3*	*24.15*	*0.09479*	*3, 6, 12, 24, 48, 90, 168, 312, 580, 1046*	*2290*	$12_2.12_2.12_2$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
7	E_7, Λ_7	**0.29530**	13.810	0.073231	**126**, 2898, 25886, 133506, 490014, 1433810, 3573054, 7902594, 15942206, 29896146	**59,400,241**	$3^{2016}.4^{5796}.5^{63}$
	D_7^+	0.26170	4.7248	**0.07273**	*64, 1092, 8064, 37842, 131328, 371940, 906816, 1976898, 3946048, 7344164*	*14,724,257*	$4^{1792}.6^{224}$
	E_7^*	0.21578	4.1872	**0.073116**	56, 938, 7688, 39746, 150248, 455114, 1171928, 2668610, 5521880, 10585514	20,601,723	$4^{1512}.6^{28}$
	D_7	0.20881	16.749	0.075686	84, 1498, 11620, 55650, 195972, 559258, 1371316, 2999682, 6003956, 11193882	22,392,919	$3^{840}.4^{2604}.5^{42}$
	A_7	0.14765	18.899	0.077396	56, 812, 5768, 26474, 91112, 256508, 623576, 1356194, 2703512, 5025692	10,089,705	$3^{336}.4^{1176}.5^{28}$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
7	D_7^*	0.07382	4.5687	0.07493	**G**: 128, 2186, 16256, 75938, 263552, 745418, 1817216, 3959426, 7902848, 14704202 $(\tau = 14)$	29,487,171	$4^{8064}.6^{64}$
	A_7^*	0.06542	3.0596	0.076187	16, 128, 688, 2746, 8752, 23536, 55568, 118498, 232976, 428752	871,661	$4^{112}.6^8$
	L_7^c	*0.11738*	**2.9000**	?	?	?	?
	$\mathbb{Z}^7$	0.03691	33.498	0.083333	14, 98, 462, 1666, 4942, 12642, 28814, 59906, 115598, 209762	433,905	$4^{84}.6^7$
	A_7^+	0.01636	*30.163*	*0.08442*	*8, 56, 224, 812, 2240, 5768, 12656, 26474, 49952, 91112*	*189,303*	6^{28}
	${}^{\mathsf{T}}A_7^*$	2.839e-3	?	?	8, 36, 120, 330, 792, 1716, 3432, 6434, 11432, 19412	43,713	$4^{20}.6^8$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
7	$\hat{A}_7^+$	3.586e-4	*137.9*	*0.1214*	*8, 14, 56, 98, 392, 644, 1400, 2198, 5096, 7532*	*17,439*	$3^{21}.12^7$
	T_7^{60}	*0.05673*	*15.87*	*0.08076*	**G***: 4, 12, 36, 100, 276, ? (*$\tau = 20$*)*	*?*	?
	S_7	*4.035e-3*	*24.15*	*0.08525*	*4, 12, 36, 100, 276, ?*	*?*	?
	V_7^{60}	*3.730e-3*	*15.00*	*0.08702*	*4, 12, 36, 100, 276, ?*	*?*	*?*
	V_7^{90}	*2.424e-3*	*32.39*	*0.09267*	*4, 12, 36, 100, 276, 724, 1676, 3592, 7012, 12868*	*26,301*	$8.8.8.8.8_2.8_2$
	Y_7^{60}	*1.652e-3*	*18.95*	*0.08854*	*3, 6, 12, 24, 48, ?*	*?*	*?*
	Y_7^{90}	*1.074e-3*	*36.73*	*0.09365*	*3, 6, 12, 24, 48, 90, 168, 312, 580, 1046*	*2290*	$12_2.12_2.12_2$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
8	E_8, E_8^*, D_8^+, Λ_8	**0.25367**	4.0587	**0.071682**	**240**, 9120, 121680, 864960, 4113840, 14905440, 44480400, 114879360, 265422960, 561403680	**1,006,201,681**	$3^{6720}.4^{21840}.5^{120}$
	D_8	0.12683	32.470	0.075914	112, 2592, 25424, 149568, 629808, 2100832, 5910288, 14610560, 32641008, 67232416	123,302,609	$3^{1344}.4^{4816}.5^{56}$
	A_8	0.08456	32.993	0.077391	72, 1332, 11832, 66222, 271224, 889716, 2476296, 6077196, 13507416, 27717948	51,019,255	$3^{504}.4^{2016}.5^{36}$
	D_8^*	0.03171	8.1174	0.074735	**G**: 256, 6560, 65280, 384064, 1614080, 5374176, 15097600, 37281920, 83222784, 171312160 $(\tau = 16)$	314,358,881	$4^{32512}.6^{128}$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
8	A_8^*	0.02969	3.6658	0.075972	*18, 162, 978, 4482, 16722, 53154, 148626, 374274, 864146, 1854882*	*3,317,445*	$4^{144}.6^9$
	L_8^c	*0.08253*	**3.1422**	?	?	?	?
	$\mathbb{Z}^8$	0.01585	64.939	0.083333	16, 128, 688, 2816, 9424, 27008, 68464, 157184, 332688, 658048	1,256,465	$4^{112}.6^8$
	A_8^+	6.599e-3	*65.99*	*0.0838*	*9, 72, 324, 1332, 4104, 11832, 28674, 66222, 136404, 271224*	*520,198*	6^{36}
	${}^{\mathrm{T}}A_8^*$	7.128e-4	?	?	9, 45, 165, 495, 1287, 3003, 6435, 12870, 24309, 43749	92,368	$4^{27}.6^9$
	$\hat{A}_8^+$	6.759e-5	*301.1*	*0.1178*	*9, 16, 72, 128, 576, 968, 2340, 3768, 9648, 14716*	*32,242*	$3^{28}.12^8$

Table 2.2 continued

n	**packing**	Δ	Θ	G	**coordination sequence** (through $k = 10$)	td_{10}	**point symbol vertex symbol**
8	T_8^{90}	*0.01585*	*64.94*	*0.08333*	**G***: 4, 12, 36, 100, 276, 724, ?* $(\tau = 16)$	*?*	?
	S_8	*9.903e-4*	*28.28*	*0.08452*	*4, 12, 36, 100, 276, 724, ?*	*?*	?
	V_8^{90}	*5.590e-4*	*49.89*	*0.09206*	*4, 12, 36, 100, 276, 724, 1908, 4390, 9876, 19682*	*37,009*	$8.8.8.8.8_2.8_2$
	Y_8^{90}	*2.327e-4*	*87.31*	*0.09266*	*3, 6, 12, 24, 48, 90, 168, 312, 580, 1046*	*2290*	$12_2.12_2.12_2$

Table 2.3: Known characteristics of selected lattices in dimension $9 \leq n \leq 24$.

n	**packing**	Δ	Θ	G	τ
9	Λ_9	**0.14577**	9.0035	*0.07206*	**272**
	D_9^+	**0.14577**	4.3331	**0.07110**	144
	D_9^*	0.01288	8.6662	0.07469	18
	A_9^*	0.01268	4.3889	0.07582	20
	A_9^5	*0.08447*	4.3402	*0.07207*	*90*
	L_9^c	*0.08149*	**4.2686**	?	?
	$\mathbb{Z}^9$	0.006442	126.81	0.08333	18
10	Λ_{10}	**0.09202**	12.409	*0.07150*	**336**
	D_{10}^+	0.07969	7.7825	**0.07081**	180
	A_{10}^*	0.005128	5.2517	0.07570	22
	L_{10}^c	*0.02995*	**5.1545**	?	?
	$\mathbb{Z}^{10}$	0.002490	249.04	0.08333	20
11	K_{11}	**0.06043**	?	?	432
	$\Lambda_{11}^{\max}$	0.05888	24.781	*0.07116*	**438**
	D_{11}^+	0.04163	8.4072	?	220
	A_{11}^*	0.001974	6.2813	0.07562	24
	A_{11}^4	*0.04740*	5.5983	*0.07025*	*132*
	L_{11}^c	*0.04124*	**5.5056**	?	?
	$\mathbb{Z}^{11}$	9.200e-4	491.40	0.08333	22
12	K_{12}, K_{12}^*	**0.04945**	17.783	**0.07010**	**756**
	$\Lambda_{12}^{\max}$	0.04173	30.419	*0.07058*	648
	D_{12}^+	0.02086	15.209	?	264
	A_{12}^*	7.271e-4	7.5101	0.07557	26
	L_{12}^c	*0.004306*	**7.4655**	?	?
	$\mathbb{Z}^{12}$	3.260e-4	973.41	0.08333	24

Table 2.3 continued.

n	**packing**	Δ	Θ	G	τ
13	K_{13}	**0.02921**	?	?	**918**
	$\Lambda_{13}^{\max}$	0.02846	*60.455*	*0.07009*	906
	A_{13}^{*}	2.569e-4	8.9768	0.07553	28
	A_{13}^{7}	?	7.8641	?	*368*
	L_{13}^{c}	*0.002255*	**7.7621**	?	?
	$\mathbb{Z}^{13}$	1.112e-4	1934.6	0.08333	26
14	Λ_{14}	**0.02162**	*98.876*	*0.06946*	**1422**
	A_{14}^{*}	8.740e-5	10.727	0.07551	30
	A_{14}^{5}	?	9.0066	?	?
	L_{14}^{c}	*0.005221*	**8.8252**	?	?
	$\mathbb{Z}^{14}$	3.658e-5	3855.6	0.08333	28
15	Λ_{15}	**0.01686**	*202.91*	*0.06892*	**2340**
	A_{15}^{*}	2.870e-5	12.817	0.07549	32
	A_{15}^{8}	?	11.602	?	?
	L_{15}^{c}	*6.206e-5*	**11.005**	?	?
	$\mathbb{Z}^{15}$	1.164e-5	7703.1	0.08333	30
16	$\Lambda_{16}, \Lambda_{16}^{*}$	**0.01471**	96.500	**0.06830**	**4320**
	A_{16}^{*}	9.116e-6	**15.311**	0.07549	34
	$\mathbb{Z}^{16}$	3.591e-6	15,422	0.08333	32
17	Λ_{17}	**0.008811**	*197.72*	*0.06822*	**5346**
	A_{17}^{*}	2.807e-6	18.288	0.07549	36
	A_{17}^{9}	?	**12.357**	?	?
	$\mathbb{Z}^{17}$	1.076e-6	30,936	0.08333	34
18	Λ_{18}	**0.005928**	*301.19*	*0.06792*	**7398**
	A_{18}^{*}	8.396e-7	**21.841**	0.07550	38
	$\mathbb{Z}^{18}$	3.134e-7	62,158	0.08333	36

Table 2.3 continued.

n	**packing**	Δ	Θ	G	τ
19	Λ_{19}	**0.004121**	*607.62*	*0.06767*	**10668**
	A_{19}^{*}	2.443e-7	26.082	0.07552	40
	A_{19}^{10}	?	**21.229**	?	?
	$\mathbb{Z}^{19}$	8.892e-8	125,077	0.08333	38
20	Λ_{20}	**0.003226**	*889.86*	*0.06731*	**17400**
	A_{20}^{*}	6.924e-8	31.143	0.07553	42
	A_{20}^{7}	?	**20.367**	?	?
	$\mathbb{Z}^{20}$	2.461e-8	252,020	0.08333	40
21	Λ_{21}	**0.002466**	*1839.5*	*0.06701*	**27720**
	A_{21}^{*}	1.914e-9	37.185	0.07555	44
	A_{21}^{11}	?	**27.773**	?	?
	$\mathbb{Z}^{21}$	6.651e-9	508,417	0.08333	42
22	Λ_{22}	**0.002128**	$\leq$ 3426.8	?	**49896**
	Λ_{22}^{*}	2.952e-4	$\leq$ **27.884**	?	1782
	A_{22}^{*}	5.168e-10	44.395	0.07558	46
	$\mathbb{Z}^{22}$	1.757e-9	1,026,792	0.08333	44
23	Λ_{23}	**0.001905**	$\leq$ 7609.0	?	**93150**
	Λ_{23}^{*}	2.788e-4	$\leq$ **15.322**	?	4600
	A_{23}^{*}	1.364e-10	53.000	0.07560	48
	$\mathbb{Z}^{23}$	4.543e-10	2,075,774	0.08333	46
24	$\Lambda_{24}, \Lambda_{24}^{*}$	**0.001930**	**7.9035**	**0.06577**	**196560**
	A_{24}^{*}	3.523e-11	63.269	0.07563	50
	$\mathbb{Z}^{24}$	1.150e-10	4,200,263	0.08333	48

2.4 Rare nonlattice packings and nets for $n \leq 8$

We now turn our attention to the problem of infinite *rare* sphere packings, with packing density *lower* than that of the corresponding cubic packing, and the closely related problem of infinite nets. For $n = 2$, this problem is essentially trivial. For $n = 3$, the richness of solutions to this problem is fascinating and, due to the intense interest in crystallographic structures with various desirable chemical properties, has been exhaustively studied and catalogued. For $n > 3$, relatively few regular constructions are known, and it appears as if what academic interest there has been has not yet led to any applications of significance in science and engineering; Bewley & Cessna (2011) intends to change this, thus motivating the present study.

Interest in n-dimensional space groups and symmetries dates back to the nineteenth century, with the work of Hessel, Bravais, Gadolin, Frankenheim, Barlow, Rodrigues, Möbius, Jordan, Sohncke, Fedorov, Schönflies, Fricke, and Klein. Historical accounts of this early work, as well as several follow-on mathematical developments related to space groups and symmetries, are available in Brown et al. (1978) and Schwarzenberger (1980). Much of the related work in the field of geometry was developed by Coxeter (1970, 1973, 1974, 1987, 1989). Despite this intense interest, there are very few explicit constructions of regular rare sphere packings for $n > 3$ available today, outside of very short treatments of the subject by O'Keeffe (1991b) and Beukemann & Klee (1992), discussed below.

As mentioned in the abstract and explored in depth in Bewley & Cessna (2011), certain emerging engineering applications now motivate the further development and deployment of quasi-infinite n-dimensional nets, with a particular focus on structured nets with low coordination number and high topological density. Such nets are well suited for the rapid spread of information in switchless computational interconnect systems with a reduced number of wires and, thus, reduced cost. In such systems, a logical network with $n > 3$ may easily be designed and built[13] and, as we will see, there are significant potential benefits for so doing. We are thus motivated to revisit the problem

[13]Recall, e.g., the "hypercube" computational interconnect system introduced several years ago; though designed with a logical network with $n > 3$, the hypercube, like most computational interconnect strategies deployed today, is significantly hampered by its inherent dependence on a Cartesian topology.

of the design of structured nets with low coordination number. Note that none of the lattice alternatives to the cubic lattice discussed in §2.3 have a coordination number lower than that of the corresponding cubic lattice, $\tau = 2n$. However, for $n = 3$, there is a wide range of stable and unstable nonlattice packings that lead to such nets; as shown below, many of these packings and nets generalize naturally to higher dimensions.

2.4.1 Net terminology

The terminology used to discuss 3D nets, most of which generalizes readily to the discussion of n-dimensional nets, has been clarified significantly over the last decade, and is now quite precise.

Recall first the measures defined in §2.2, including the *coordination number*, the *coordination sequence*, and a k-hop measure of *local topological density* given by the cumulative sum of all nodes reached within k hops from origin, denoted td_k (Tables 2.1 and 2.2 list this quantity for $k = 10$). O'Keeffe (1991a) defines another, sometimes preferred (see, e.g., Grosse-Kunstleve et al. 1996) measure of *global topological density*, $td = \lim_{k\to\infty} td_k/k^n$, which reveals the rate of growth of td_k with k in the limit of large k. [For a uninodal n-dimensional net, td may be found by representing[14] the coordination sequence as an $(n-1)$'th-order polynomial in the number of hops k, then taking the leading coefficient of this polynomial and dividing by n.] Despite some impressive efforts in representing coordination sequences with such polynomials (see, e.g., Conway & Sloane 1997, and the references contained therein), the measure td is currently unknown for most of the nets discussed here. As a matter of computational tractability, we thus resort in the present work to the tabulation of the local topological density measure, td_{10}, as this measure is much easier to compute.

Our attention in this work is focused almost exclusively on *equilibrium packings* (that is, on sphere packings which, if unperturbed, can bear compressive loads applied at the edges of a packing that is built out to fill a finite convex domain) and their corresponding *equilibrium nets* (which are constructed with tensile members connecting nearest-neighbor nodes, and can bear tensile loads applied at the edges of a finite con-

[14] Or by *approximating* this coordination sequence as an $(n-1)$'th-order polynomial for large k, if such a polynomial does not fit exactly.

vex domain)[15,16]. Equilibrium packings fall into two catagories: stable (that is, sphere packings which, if perturbed, oscillate about their equilibrium configurations, and return to these configurations if there is damping present in the system) and unstable (that is, sphere packings which depart from equilibrium if perturbed); we consider both.

After years of conflicting terminology in the literature on nets, the concepts of *cycles*, *rings*, *strong rings*, *tilings*, *natural tilings*, *point symbols*, and *vertex symbols* have, in 3D, finally crystallized. The reader is referred to Blatov et al. (2009) and the references contained therein for description of this modern terminology, as well as numerous cautions concerning the conflicting nomenclatures adopted elsewhere in the published literature. In short:

- A *cycle* is a sequence of nodes in a net, connected by edges, such that the first and last nodes of the sequence coincide, while none of the other nodes in the sequence appears more than once.
- A *cycle sum*, of cycles A and B, is the union of those edges in either A or B but not both.
- A *ring* is a cycle that is not the sum of two smaller cycles.
- A *strong ring* is a cycle that is not the sum of any number of smaller cycles.
- A *tiling* of $\mathbb{R}^3$ by a 3D net is simply the dissection of 3D space into volumes whose faces, which in general may be curved (as *minimal surfaces*, like soap bubbles; see, e.g., Sadoc & Rivier 1999), are bounded by cycles of the net. A 3D net generally admits many tilings.
- The *dual* of a tiling is the unique new tiling obtained by placing a new vertex inside each original tile and connecting the vertices of adjacent tiles (that is, with shared faces) in the original tiling with edges. If a tiling and its dual are identical, the tiling is said to be *self-dual*. The dual of a dual is the original tiling.

[15] A family of structures with both tensile and compressive members, known as *tensegrity*, might be said to cover the gap between purely compressive sphere packings and purely tensile nets. The mathematical characterization of tensegrity systems in 3D is now precise, due largely to the work of Skelton & de Oliveira (2009). An interesting extension of the present study would be to generalize such tensegrity systems to $n > 3$ dimensions.

[16] For the purpose of the applications studied in Chapters 3 and 4, we do not actually use the property of mechanical equilibrium of the corresponding structure; this property may rather be considered as a convenient means to an end when designing a regular packing or net. Several nets discussed in the literature (see, e.g., Wells 1977, page 80) are not equilibrium sphere packings, and might be interesting to consider further.

- A *natural tiling* of $\mathbb{R}^3$ by a 3D net is a tiling with the smallest possible tiles such that the tiles have the maximum combinatorial symmetry and all the faces of the tiles are strong rings. A 3D net often[17] admits a unique natural tiling. If a tiling and its dual are both natural, the pair is referred to as *natural duals*. If a natural tiling is self-dual, it is said to be *naturally self-dual*.
- The *point symbol* of a uninodal net, of the form $A^a.B^b.C^c\ldots$, indicates that there are a pairs of edges touching the node at the origin with shortest cycles of length A, b pairs of edges touching the node at the origin with shortest cycles of length B (with $B > A$), etc. Note that the sum of the superscripts in a point symbol totals $\tau(\tau-1)/2$.
- The *vertex symbol* of a uninodal net, of the form $A_a.B_b.C_c\ldots$, indicates that the first pair of edges touching the node at the origin has a shortest rings of length A, the second pair of edges touching the node at the origin has b shortest rings of length B, etc. If for any entry there is only 1 such shortest ring, the subscript is suppressed; if for any entry there is no ring, a subscript $*$ is used. The entries are sorted such that smaller rings are listed first, and when two rings of the same size appear, the entry with the smaller subscript is listed first. In the special case of $\tau = 4$, the six entries of the vertex symbol are listed as three pairs of entries, with each pair of entries corresponding to opposite pairs of edges, and are otherwise again sorted from smallest to largest. Note that the number of entries in a vertex symbol is $\tau(\tau-1)/2$.

The concepts of *cycles*, *rings*, *strong rings*, *point symbols*, and *vertex symbols* extend immediately to n dimensions; for practical considerations (specifically, because the number of entries in a vertex symbol gets unmanageable for large τ), we list the point symbol in Table 2.2 wherever $\tau \geq 5$, and the vertex symbol where $\tau \leq 4$. The extension of the tiling concept to n dimensions is more delicate, and is discussed further in §2.4.5.

Following Delgado-Friedrichs et al. (2003a,b), the *regularity* of a 3D net may now be characterized precisely. In short, consider a 3D net with p kinds of vertex and q kinds of edge and whose natural tiling is characterized by r kinds of face and s kinds of tile. Delgado-Friedrichs & Huson (2000) introduced a clear and self-consistent method for characterizing the regularity of such a net simply by forming the array *pqrs*:

[17] Unfortunately, not all 3D nets have natural tilings, and some have multiple natural tilings; §3 of Blatov et al. (2007) discusses this issue further.

examining the 4-digit number so formed, referred to as the *transitivity* of the net, the most "regular" 3D nets are generally those with the smallest transitivity.

Finally, a *minimal net* is a net with the minimum possible number of vertices and edges in its primitive cell[18]; that is, upon deletion of any further edges in the primitive cell, the resulting net breaks into multiple disconnected structures. Beukemann & Klee (1992) establish that there are only 15 such minimal nets in 3D. Delgado-Friedrichs & O'Keeffe (2003) define a 3D net as *barycentric* if every vertex is placed in the center of gravity of its neighbors (to which it is connected by edges). Bonneau et al. (2004), in turn, establish that 7 of the 15 such minimal nets in 3D have *collisions*; that is, when arranged in barycentric fashion, the location of two or more vertices coincide (and, thus, the net is in a sense degenerate). Of the 8 remaining minimal nets without collision, five are uninodal.

2.4.2 2D nets

Consider first the development of uninodal 2D nets with low coordination number. Start from the triangular ($A_2^* \cong A_2$) lattice (see §2.2) and perform a red/black/blue coloring of the nodes such that no two nearest-neighbor nodes are the same color. If we retain only the red and black nodes, we are left with the *honeycomb packing* (see Figure 2.1e), and the corresponding net is an array of hexagons. The coordination number of this stable sphere packing is $\tau = 3$, which is less than that of the 2D square packing ($\tau = 4$); this implies fewer wires in the corresponding computational interconnect application. Unfortunately, the topological density of this net is quite poor, with $td_{10} = 166$ (that is, with information spreading from one node to only 165 others after a message progresses 10 hops in the network application). We are thus motivated to explore other ways of constructing structured (that is, easy-to-build and easy-to-navigate) nets with low coordination number (that is, with low cost) but high topological density (that is, with a fast spread of information).

Note that the honeycomb packing has a packing density which is less than that of the corresponding triangular and square lattices discussed previously (see Table 2.2).

[18] A *primitive cell* of a net is the smallest fundamental volume (e.g., hypercube) that, when repeated as an infinite array in all directions with zero spacing, generates the net.

If minimization of packing density is the goal[19], then the honeycomb packing may be *augmented* by replacing every sphere with a set of three spheres in contact, each such set forming an equilateral triangle which touches the neighbors in exactly the same locations as the single sphere which it replaces in the original (unaugmented) packing (see, e.g., Heesch & Laves 1933, Figure 13). The packing density of the resulting stable *augmented honeycomb* packing is less than $2/3$ that of the original honeycomb packing (see Table 2.2), and is the rarest uninodal sphere packing available in 2D.

2.4.3 A List of Twelve "highly regular" uninodal 3D nets

There are far too many 3D nets to review them all here. We thus identify a List of Twelve highly "regular" (as defined in §2.4.1, via their transitivity) uninodal 3D nets upon which we will focus our attention and which, following Delgado-Friedrichs et al. (2003a,b), we denote (listing from dense to rare):

1. **fcu**: face-centered cubic (FCC),
2. **bcu**: body-centered cubic (BCC),
3. **pcu**: cubic,
4. **qtz**: quartz,
5. **nbo**: NbO,
6. **dia**: diamond,
7. **sod**: sodalite,
8. **qzd**: quartz dual,
9. **cds**: $CdSO_4$,
10. **bto**: B_2O_3,
11. **ths**: $ThSi_2$,
12. **srs**: $SrSi_2$.

See Table 2.2 for the common names, associated packings, and key characteristics of each[20]. These twelve nets have been studied thoroughly in the literature, including the landmark work of Wells (1977, 1979, 1983, 1984) and scores of important publications since, including Koch & Fischer (1995, 2006) and the numerous references contained therein; space does not allow a comprehensive review of this broad body of literature here, nor even a comprehensive analysis of these twelve well-studied nets. Suffice it to say here that included in our List of Twelve are the 5 *regular* nets (that is, of transitivity 1111), **bcu**, **pcu**, **nbo**, **dia**, and **srs**, and the 1 *quasiregular* net (of transitivity 1112), **fcu**, as well as 2 of the 14 *semiregular* nets (of transitivity 11*rs*), **qtz** and **sod** (both of which have transitivity 1121), as discussed in Delgado-Friedrichs et al. (2003a,b). Also included in this list are the 5 uninodal minimal nets without collision, **pcu**, **dia**, **cds**, **srs**, and **ths**, the first 4 of which are naturally self-dual, as discussed in Bonneau et al. (2004,

[19] Note that, for $n > 3$, the authors are actually unaware of any practical application, other than mathematical curiosity, for which minimization of packing density is a significant goal.

[20] Again, clear drawings of each of these nets are available at `http://rcsr.anu.edu.au/nets/fcu`, where "`fcu`" may be replaced by any of the lowercase boldface three-letter identifiers given here.

Table 1); note that **cds** is of transitivity 1221, and **ths** is of transitivity 1211[21]. The remaining 2 nets on our List of Twelve, **qzd** (transitivity 1211; see Delgado-Friedrichs et al. 2003c) and **bto** (transitivity 1221; see Blatov 2007), are included because of their close structural relationship to the others, as discussed further in §2.4.4. We also note that four on our List of Twelve, **qtz**, **qzd**, **bto**, and **srs**, are *chiral* (that is, these nets twist in such a way that the nets and their reflections are not superposable).

The 12 remaining semiregular nets (of transitivity 11*rs*) of Delgado-Friedrichs et al. (2003b, Table 1) are the next natural candidates in this taxonomy (**hxg**, **crs**, **reo**, and **rhr** might be of particular interest), perhaps followed by the 28 binodal edge-transitive nets (of transitivity 21*rs*) of Delgado-Friedrichs et al. (2006, Table 1) and the 3 binodal minimal nets without collision (of transitivity 2222, 2211, and 2321) of Bonneau et al. (2004, Table 1). Note that just half of the List of Twelve considered here (specifically, in order of frequency, **dia**, **pcu**, **srs**, **ths**, **nbo**, and **cds**) account for 66% of the 774 uninodal metal-organic frameworks (MOFs) tabulated in the Cambridge Structural Database (CSD), as reviewed by Ockwig et al. (2005), thus indicating the prevalence in nature of several of the structures considered here.

The idea of augmentation, introduced in §2.4.2, extends directly to many 3D nets in order to reduce packing density. For example, in the (stable) packings related to the **dia** and **sod** nets (discussed further in §2.4.4 and §2.4.4 respectively), both of which have coordination number 4, we may replace each sphere with a set of four spheres in contact, each such set of spheres forming a tetrahedron, creating what is referred to as the *augmented diamond* (**dia-a**) and *augmented sodalite* (**sod-a**) nets. In the case of the augmentation of the packing related to the **dia** net, each tetrahedral set touches the neighbors in exactly the same locations as the single sphere which it replaces in the original (unaugmented) packing (see Heesch & Laves 1933, Figure 12). In the case of the augmentation of the packing related to the **sod** net, as the angles between the 4 nearest neighbors of any node are not uniform in the **sod** net, each tetrahedral set is slightly larger than the single sphere which they replace in the original (unaugmented) packing, and the contact points are slightly shifted (O'Keeffe 1991b); note that the packing associated with the **sod-a** net is the rarest uninodal stable 3D packing currently known. On

[21] As illustrated in Bonneau et al. (2004, Figure 3), a self-dual tiling of **ths** may in fact be constructed; this tiling has transitivity 1221.

the other hand, in the augmentation of the (unstable) packing related to the **srs** net, which has coordination number 3, we may replace each sphere with a set of three spheres in contact, each such set of spheres, as in the augmentation of the honeycomb packing, forming an equilateral triangle and touching the neighbors in exactly the same locations as the single sphere which it replaces in the original (unaugmented) packing (see Heesch & Laves 1933, Figure 10); note that the packing associated with the resulting **srs-a** net is the rarest uninodal unstable 3D packing known.

Comparing augmented honeycomb to honeycomb, **dia-a** (transitivity 1222) to **dia**, **sod-a** (transitivity 1332) to **sod**, and **srs-a** (transitivity 1221) to **srs**, it is seen that augmentation, while reducing the packing density Δ (see Table 2.2), also significantly reduces both the topological density, td_{10}, and the regularity of the resulting net. Thus, the process of augmentation appears to be of little interest for the purpose of developing efficient computational interconnects. Note that Fischer (2005) and Dorozinski & Fischer (2006) show that the process of augmentation can be repeated indefinitely in order to obtain (non-uninodal) sphere packings of arbitrarily low packing density.

Finally, there are two other 3D nets which, though less regular than our List of Twelve, are worthy of "honorable mention": *hexagonal close packing* (**hcp**, transitivity 1232) and *lonsdaleite* (**lon**, transitivity 1222). As hinted by their identical packing densities (see Table 2.2a), **hcp** is closely related to **fcu**, and **lon** is closely related to **dia**; curiously, both have slightly *higher* values of td_{10} than do their more regular cousins. The relations between these two pairs of packings is readily evident when they are considered as built up in layers, as introduced in the second paragraph of §2.3.4 and discussed further below.

The A_3 lattice (a.k.a. FCC, corresponding to the **fcu** net) may be built up as an alternating series of three 2D triangular (A_2) layers, offset from each other in the form *abcabc*..., with the nodes in one layer over the holes in the layer below; **hcp** is built up similarly, but with two alternating layers, offset from each other in the form *abab*...

Similarly, the sphere packings corresponding to the **dia** and **lon** nets may be built up as alternating series of approximately 2D honeycomb layers offset from each other. These honeycomb "layers" are in fact not quite 2D; if the nodes in a single layer are marked with an alternate red/black coloring, the red nodes are raised a bit and the black

nodes lowered a bit. In both packings, the layers are offset from each other, with the lowered nodes in one layer directly over the raised nodes in the other. In the packing corresponding to the **dia** net, there are three such alternating layers stacked in the form *abcabc*...; in the packing corresponding to the **lon** net, there are two such alternating layers stacked in the form *abab*...

2.4.4 Uninodal extension of several regular 3D nets to higher dimensions

The **fcu** net is based on the $D_3 \cong A_3$ lattice, and thus may be extended to n dimensions in two obvious ways (that is, via A_n or D_n). The **bcu** net is based on the $D_3^* \cong A_3^*$ lattice, and thus may also be extended to n dimensions in two obvious ways (via A_n^* or D_n^*). The **pcu** net is based on the $\mathbb{Z}^3$ lattice, and thus extends to n dimension via $\mathbb{Z}^n$. This section explores how most of the other nets on the List of Twelve described above extend naturally to higher dimensions.

It is important to recall that the nets in the D_n^* case for $n > 4$ turn out to be a bit peculiar, as discussed further in §2.3.3; the T_n^{90} and T_n^{60} nets encountered in §2.4.4 are similar.

Extending dia: the A_n^+ and D_n^+ packings

The **dia** net may be obtained from the well-known D_3^+ packing defined in (2.6) (see also Sloane 1987), and thus extends naturally to n dimensions as D_n^+. However, there is an alternative construction of the **dia** net, described below and denoted A_n^+, which is equivalent to D_n^+ for $n = 3$ but extends to n dimensions differently. In fact, a third extension of the **dia** net to n dimensions, the V_n^{90} construction, is introduced in §2.4.4. These alternative extensions of the **dia** net to n dimensions, with low coordination number, are perhaps better suited than D_n^+ for many practical applications. We thus stress that names such as "n-dimensional diamond" are parochial, as there are sometimes multiple "natural" n-dimensional extensions of a net related to a given three-dimensional crystalline structure (e.g., D_n^+, A_n^+, and V_n^{90}). For n-dimensional nets in general, we thus strongly prefer names derived from a corresponding well-defined n-dimensional lattice

or, when such a name is not available, names evocative of their n-dimensional construction; this preference is in sharp contrast with the names suggested by O'Keeffe (1991b).

Recall the first paragraph of §2.4.2. Now start from a BCC ($A_3^* \cong D_3^*$) lattice and perform a red/black/blue/yellow coloring of the points such that no two nearest-neighbor points are the same color. If we retain only the red and black points, we are left with the diamond packing. The coordination number of this packing is $\tau = 4$, which is less than that of the 3D cubic packing ($\tau = 6$), but also has a reduced topological density, as quantified by td_{10} (see Table 2.2). The diamond packing also has a packing density which is less than that of the corresponding FCC, BCC, and cubic lattices.

Note in general [see (2.8a)] that A_n^* may be defined as the union of $n+1$ shifted A_n lattices, which is analogous to the property [see (2.5a)] that D_n^* may be defined as the union of 4 shifted D_n lattices. Recall from (2.6) that D_n^+, which we referred to the *offset checkerboard packing*, was defined as the union of just 2 shifted D_n lattices, and generates the diamond packing in 3D (where $D_3 \cong A_3$). Motivated by the previous paragraph and the first paragraph of §2.4.2, we are thus also keenly interested in the nonlattice packing considered in Table 1 of O'Keeffe (1991b), denoted here A_n^+ and referred to as the *offset zero-sum packing*, and which is defined as the union of just 2 shifted A_n lattices [cf. (2.6), (2.8)]:

$$A_n^+ = A_n \cup ([1] + A_n), \quad \text{where} \quad [1]_k = \begin{cases} \frac{1}{n+1} & k \leq n, \\ \frac{-n}{n+1} & k = n+1. \end{cases} \tag{2.15}$$

The coordination number of the regular uninodal nonlattice packing A_n^+ is $n+1$, with these $n+1$ nearest neighbors forming an n-dimensional *simplex* [that is, in n dimensions, a polytope with $n+1$ vertices—e.g., in $n = 3$ dimensions, a tetrahedron (with 4 vertices)]. The generalization of the honeycomb and diamond packings to higher dimensions given by A_n^+ is significant, as it illustrates how a highly regular stable packing with coordination number lower than that of the cubic lattice may be extended to dimension $n > 3$. Note also that the nonlattice packings A_n^+ are distinct from the lattice packings A_n^r defined in (2.9), which are generated in a similar manner.

Augmenting the A_n^+ packing: $\hat{A}_n^+$

The third paragraph of §2.4.3 discusses the augmentation of the A_3^+ packing, replacing each sphere with a tetrahedral set of 4 smaller spheres. This idea extends immediately to the augmentation, in n dimensions, of the A_n^+ packing discussed above, replacing each (n-dimensional) sphere with an n-dimensional simplex of $n+1$ smaller spheres.

Extending sod: the ${}^{\mathsf{T}}A_n^*$ packing

The familiar **sod** net is formed by the edges of the Voronoï tesselation of space formed by the A_3^* (that is, BCC) packing, with the nodes of the net located at the *holes* of the packing rather than at the centers of the spheres of the packing. As noted by O'Keeffe (1991b), this construction extends immediately to the n-dimensional net formed by the Voronoï tesselation of space via the A_n^* packing. Constructing the A_n^* packing as defined in §2.3.4, the holes of this packing that are nearest to the origin (that is, in its Voronoï tesselation, the corners of the Voronoï cell which contains the origin) are given by the $(n+1)!$ permutations of the vector (see Conway & Sloane, 1999, page 474):

$$\frac{1}{2(n+1)}\begin{pmatrix} -n & -n+2 & -n+4 & \dots & n \end{pmatrix}^T .$$

These nodal points [which, like the lattice points of A_n^* itself, are defined in an $(n+1)$-dimensional space, but all lie in the n-dimensional subspace orthogonal to the vector $\mathbf{n}_{A_n}$ defined in (2.7b)] are equidistant from their $n+1$ nearest neighbors, and form *permutohedra* (in 3D, *truncated octahedra*) which tile n-dimensional space. Note that these nodal points themselves form a uninodal sphere packing with coordination number $\tau = n+1$; due to its relationship to the *tesselation* of space via the points of the A_n^* packing, we thus introduce the notation ${}^{\mathsf{T}}A_n^*$ for this packing.

Extending nbo: the S_n construction

The **nbo** net, a subset of the **pcu** net, has an obvious uninodal extension to n dimensions with $\tau = 4$, which may be visualized as the contact graph formed by repeating a unit hypercube pattern as an infinite array with unit spacing (see Figure 2.11),

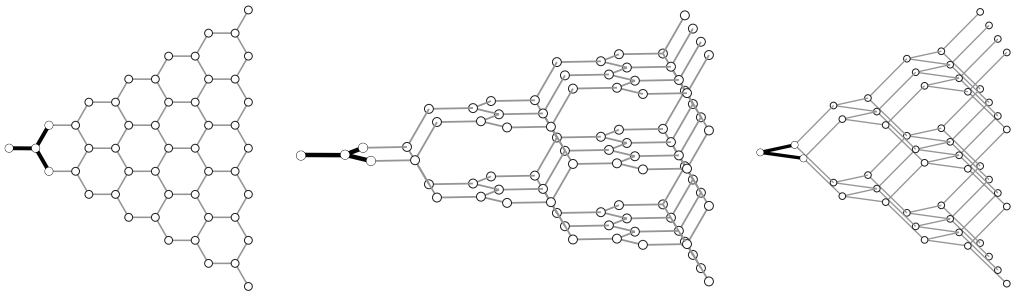

Figure 2.7: Construction of three rare packings: (left) the Y_2 (honeycomb) net, (center) the Y_3^{90} (**ths**) net, and (right) the V_3^{90} (**dia**) net. All three constructions build from left to right in the above figures from a basic "Y" or "V" stencil, and have obvious extensions to higher dimensions.

where each hypercube itself has two paths which "snake" along the edges from the $(0,0,\cdots,0,0)$ node to the $(1,1,\cdots,1,1)$ node, one coordinate direction at a time; we thus suggest the symbol S_n to denote this construction. These two paths touch at the opposite corners of the unit hypercube.

Extending ths and bto: the Y_n^{90} and Y_n^{60} constructions

The honeycomb packing A_2^+, of coordination number $\tau = 3$, contains a fundamental Y-shaped stencil. As illustrated in Figure 2.7a, starting with this Y stencil and adjoining translates of itself, tip to tip, builds up the honeycomb packing in 2D. Extending this idea to 3D, as illustrated in Figure 2.7b, we may "twist" the Y stencil by 90° at each level: starting with the basic Y stencil in, say, the $\mathbf{e}^1$-$\mathbf{e}^2$ plane, we can shift to the right (in $\mathbf{e}^1$) and adjoin Y stencils twisted by 90° (that is, aligned in the $\mathbf{e}^1$-$\mathbf{e}^3$ plane), then shift to the right again and adjoin Y stencils twisted again (aligned in the $\mathbf{e}^1$-$\mathbf{e}^2$ plane), etc. This construction forms the **ths** net in 3D, and extends immediately to dimension $n > 3$; we thus denote this construction Y_n^{90}.

Instead of twisting the Y stencil by 90° at each step, we may instead twist it by 60°. This forms the **bto** net in 3D. As with the **hcp** versus **fcu** and **lon** versus **dia** nets in 3D, as described at the end of §2.4.3, there is a bit of flexibility in terms of the ordering of the the successive twists for $n > 3$. A highly regular net for odd n, which we denote Y_n^{60}, is formed by pairing off the dimensions after the first and alternating the

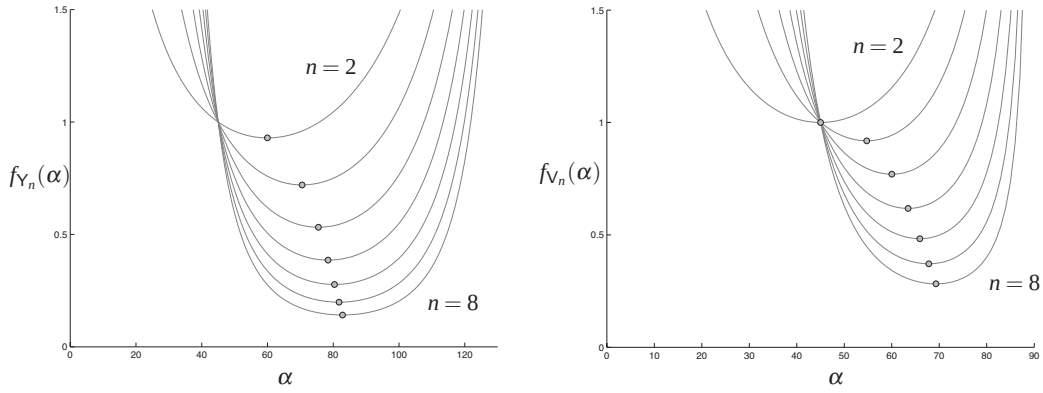

Figure 2.8: Variation of the Voronoï volume of the Y_n^{90} & Y_n^{60} and (right) V_n^{90} & V_n^{60} packings as a function of α for $n=2$ to $n=8$.

twists as follows: starting with the basic Y stencil in, say, the $\mathbf{e}^1$-$\mathbf{e}^2$ plane, we continue by adjoining Y stencils in the $\mathbf{e}^1$-$\mathbf{e}^4$ plane, then in the $\mathbf{e}^1$-$\mathbf{e}^6$ plane, etc. We then adjoin Y stencils in the $\mathbf{e}^1$-$\mathbf{z}_{23}^{60}$ plane, where $\mathbf{z}_{23}^{60}$ is the vector formed by rotating the $\mathbf{e}^2$ unit vector 60° in the direction towards $\mathbf{e}^3$; we continue by adjoining Y stencils in the $\mathbf{e}^1$-$\mathbf{z}_{45}^{60}$ plane, then in the $\mathbf{e}^1$-$\mathbf{z}_{67}^{60}$ plane, etc. Next, we adjoin Y stencils in the $\mathbf{e}^1$-$\mathbf{z}_{23}^{120}$ plane, where $\mathbf{z}_{23}^{120}$ is the vector formed by rotating the $\mathbf{z}_{23}^{60}$ vector 60° further in the $\mathbf{e}^2$-$\mathbf{e}^3$ plane; we continue by adjoining Y stencils in the $\mathbf{e}^1$-$\mathbf{z}_{45}^{120}$ plane, then in the $\mathbf{e}^1$-$\mathbf{z}_{67}^{120}$ plane, etc., and repeat (that is, with stencils again aligned in the $\mathbf{e}^1$-$\mathbf{e}^2$ plane).

The Y_n^{90} and Y_n^{60} constructions have a parameter, denoted α and defined as half of the angle between the two top branches of the Y stencil (thus, $\alpha \to 0^\circ$ closes down the Y to an I, whereas $\alpha \to 90^\circ$ opens up the Y to a T). The Voronoï volume of the Y_n^{90} and Y_n^{60} constructions may be written as simple functions of α as follows:

$$\left.\begin{aligned} \mathscr{V}_{\mathsf{Y}_n^{90}}(\alpha) &= f_{\mathsf{Y}_n}(\alpha)\,\mathscr{V}_{\mathsf{Y}_n^{90}}(\bar{\alpha}) \\ \mathscr{V}_{\mathsf{Y}_n^{60}}(\alpha) &= f_{\mathsf{Y}_n}(\alpha)\,\mathscr{V}_{\mathsf{Y}_n^{60}}(\bar{\alpha}) \end{aligned}\right\} \quad \text{with} \quad \bar{\alpha} = 45^\circ.$$

Where $f_{\mathsf{Y}_n}(\alpha) = (2-\sqrt{2})\,(1+\cos\alpha)\,(\sqrt{2}\sin\alpha)^{n-1}$. This relation is plotted in Figure 2.8a. The characteristics of Y_n^{90} and Y_n^{60} reported in Table 2.2 are computed for $\alpha = \cos^{-1}(1/n)$, as marked with circles in Figure 2.8a, which maximizes the Voronoï volume and, thus, minimizes the packing density. An alternative natural choice is $\alpha = 60$, which results in barycentric constructions of Y_n^{90} and Y_n^{60}.

Extending dia and qtz: the V_n^{90} and V_n^{60} constructions

The V_n^{90} and V_n^{60} constructions are defined in an identical manner as their Y_n^{90} and Y_n^{60} counterparts, with a V stencil replacing the Y stencil (see, e.g., Figure 2.7c), thus resulting in nets with coordination number $\tau = 4$ instead of $\tau = 3$. These constructions lead to the **dia** and **qtz** nets in 3D.

As with the Y_n^{90} and Y_n^{60} construction, the V_n^{90} and V_n^{60} constructions have a parameter, denoted α and defined as half of the angle between the two top branches of the V stencil. The Voronoï volume of the V_n^{90} and V_n^{60} constructions may be written as simple functions of α as follows:

$$\left.\begin{aligned} \mathscr{V}_{\mathsf{V}_n^{90}}(\alpha) &= f_{\mathsf{V}_n}(\alpha)\,\mathscr{V}_{\mathsf{V}_n^{90}}(\bar{\alpha}) \\ \mathscr{V}_{\mathsf{V}_n^{60}}(\alpha) &= f_{\mathsf{V}_n}(\alpha)\,\mathscr{V}_{\mathsf{V}_n^{60}}(\bar{\alpha}) \end{aligned}\right\} \quad \text{with} \quad \bar{\alpha} = 45^\circ, \quad f_{\mathsf{V}_n}(\alpha) = 2^{n/2}\cos\alpha\,(\sin\alpha)^{n-1}.$$

This relation is plotted in Figure 2.8b. The characteristics of V_n^{90} and V_n^{60} reported in Table 2.2 are computed for $\alpha = \cos^{-1}(1/\sqrt{n})$, as marked with circles in Figure 2.8a, which maximize the Voronoï volumes and, thus, minimize the packing density. Note that the V_n^{90} and V_n^{60} constructions are barycentric for any α in the range $0 < \alpha < 90^\circ$.

Extending cds and qzd: the T_n^{90} and T_n^{60} constructions

The T_n^{90} and T_n^{60} constructions are defined in an analogous manner as their Y_n^{90}, V_n^{90}, Y_n^{60}, and V_n^{60} counterparts, and lead to the **cds** and **qzd** nets in 3D. The only difference now is that, instead of adjoining two new Y or V symbols on the tips of each Y or V symbol in the previous layer, we now adjoin a single new T symbol centered atop each T symbol in the previous layer, appropriately twisted; these constructions thus result in nets with coordination number $\tau = 4$. Note that the "horizontal" and "vertical" distances between nodes in these constructions are equal, and that these constructions are parameter free and barycentric.

Note that the x_1 direction is special in the Y_n^{90}, Y_n^{60}, V_n^{90}, V_n^{60}, T_n^{90}, and T_n^{60} constructions. These constructions are configured in this way intentionally, in order to construct equilibrium packings; however, other variations are certainly possible. Note also that the Y_n^{60}, V_n^{60}, and T_n^{60} constructions involve pairing off the dimensions after the first and rotating in each pair of dimensions 60° at a time, in the manner described in

§2.4.4. If we follow the same procedure but rotate 90° at a time, we recover nets equivalent to the corresponding Y_n^{90}, V_n^{90}, and T_n^{90} nets, respectively, as defined previously.

Note also that the Y_n^{90}, V_n^{90}, and T_n^{90} constructions form square layers in the $\mathbf{e}_2$-$\mathbf{e}_3$ plane, the $\mathbf{e}_4$-$\mathbf{e}_5$ plane, the $\mathbf{e}_6$-$\mathbf{e}_7$ plane, etc., whereas the Y_n^{60}, V_n^{60}, and T_n^{60} constructions form triangular layers in these planes. In the resulting Y_n^{90}, Y_n^{60}, V_n^{90}, and V_n^{60} nets, there are, in fact, no edges of the net within these layers (that is, all of the edges connect nodes in different layers). On the other hand, in the resulting T_n^{90} and T_n^{60} nets, each node is connected via edges of the net to exactly two others (note: *not* four or six) within these layers. As with the peculiar D_n^* net discussed previously, the T_n^{90} and T_n^{60} constructions are, in fact, *not* contact graphs of the corresponding sphere packings[22]; some bonds must be cut in the corresponding contact graphs (which, in the case of T_n^{90}, is simply $\mathbb{Z}^n$) in order to form the T_n^{90} and T_n^{60} nets.

Other extensions

Sections 2.4.4 through 2.4.4 summarize several uninodal families of n-dimensional extrapolations of some common 3D nets; most of these (unless indicated otherwise, via references to existing literature) are new. Note that O'Keeffe (1991b) mentions two other such extensions, one corresponding to the **lon** net and one corresponding to the **sod-a**, the latter of which is currently the rarest uninodal stable packing known for $n > 3$ (and which, consistent with the above developed naming conventions, we might suggest to identify as ${}^{\mathrm{T}}\hat{A}_n^*$). Beukemann & Klee (1992, page 50) mentions two extensions of their own (at least, to $n = 4$), both related to the **dia** net. Judging from the vast assortment of distinct rare sphere packings and related nets available in 3D, there are certainly *many* more uninodal extensions to higher dimensions of regular rare 3D packings that are still awaiting discovery; we have focused our attention here on what appear to be several of the most regular. The regularity of n-dimensional nets for $n > 3$ is discussed further below.

[22]Note that there is a lower-symmetry form of **cds** in 3D with four nearest neighbors per node whose contact graph does generate the **cds** net; see Delgado-Friedrichs (2005, Figure 1). Lower symmetry forms of other T_n^{90} and T_n^{60} constructions, whose nets are contact graphs, might also exist.

2.4.5 Regularity and transitivity of n-dimensional nets for $n > 3$

As reviewed in §2.4.1, the regularity of a 3D net is defined based on its transitivity, which in turn is based on the so-called natural tiling of the 3D net. The natural tiles of 3D nets have been thoroughly characterized in the literature for all of the most regular 3D nets available. In §2.4.4, we described uninodal extensions of several regular 3D nets to higher dimensions, and mentioned that many more such uninodal nets with $n > 3$ most certainly exist. The natural question to ask, then, is how the concepts of regularity and transitivity can be extended to higher dimensions, so that we may differentiate between these nets and identify those which are the most regular.

This question is difficult to visualize in dimensions higher than three, and requires a symbolic/numerical description of the net to proceed. The net arising from the $\mathbb{Z}^n$ lattice for $n = 4, 5, \ldots$, which is naturally tiled by n-dimensional hypercubes, is by far the easiest starting point. Denote first the symbols $\{v, w, x, y, z\}$ as variables that range from 0 to 1. The 3D unit cube, denoted $\{xyz\}$, has six faces, $\{xy0, xy1, x0z, x1z, 0yz, 1yz\}$. Each face, in turn, has four edges; e.g., $\{0yz\}$ has edges $\{0y0, 0y1, 00z, 01z\}$. Finally, each edge connects two nodes; e.g., $\{00z\}$ connects nodes $\{000, 001\}$. The 4D unit hypercube, $\{wxyz\}$, has eight 3-faces, $\{wxy0, wxy1, wx0z, wx1z, w0yz, w1yz, 0xyz, 1xyz\}$, each 3-face has six 2-faces, each 2-face has four edges, and each edge connects two nodes. The 5D unit hypercube, $\{vwxyz\}$, has ten 4-faces, each 4-face has eight 3-faces, each 3-face has six 2-faces, each 2-face has four edges, and each edge connects two nodes; etc.

In 3D, as reviewed in §2.4.1, the transitivity is based on the number of distinct nodes, edges, (two-dimensional) faces, and (three-dimensional) tiles. By analogy, then, in 4D we may define the transitivity of a net based on the number of distinct nodes, edges, 2-faces, 3-faces, and (4-dimensional) tiles in the natural tiling. Similarly, in 5D, we may define the transitivity based on the number of distinct nodes, edges, 2-faces, 3-faces, 4-faces and (5-dimensional) tiles in the natural tiling; etc. Via this definition, the net derived from the $\mathbb{Z}^4$ lattice has transitivity 11111, the net derived from the $\mathbb{Z}^5$ lattice has transitivity 111111, etc.

For all of the other nets with $n > 3$ listed in Table 2.2, the computation of the transitivity remains an important unsolved problem. Note that, in a tiling corresponding

to a 3D net, the (two-dimensional) faces of the (three-dimensional) tiles are, in general, minimal surfaces stretched over non-planar frames built from (one-dimensional) edges between several nodal points defined in 3D. In a tiling corresponding to an n-dimensional net for $n > 3$, the 2-faces of the tiles are, in general, minimal surfaces stretched over nonplanar frames between several nodes defined in n dimensions. [Note that the computation of such minimal surfaces in n dimensions is straightforward using standard level set methods; see, e.g., Cecil (2005).] Several of these nonplanar 2-faces combine to form the boundaries of each 3-face, which itself is not confined to lie within a 3D subspace of the n-dimensional domain. Several of these 3-faces then combine to form the boundaries of each 4-face; etc.

Identification of such high-dimensional natural tilings is apparently a task that could be readily accomplished numerically, but is, in general, expected to be difficult to visualize.

2.5 Coding theory

Though the lattices that arise from n-dimensional sphere packings have connections that permeate many foundational concepts in number theory and pure geometry, the list of successful direct applications in science and engineering of n-dimensional sphere packings with $n > 3$ is currently surprisingly short[23]; this list includes

- the numerical evaluation of integrals (Sloan & Kachoyan 1987),
- the solution of the linear Diophantine inequalities that arise in integer linear programming (Schrijver 1986),
- the characterization of crystals with curious five-fold symmetries (Janssen 1986),
- attempts at unifying the 4 fundamental forces (in 10, 11, or 26 dimensions) via superstring theory (Kaku 1999), and
- the development of maximally effective numerical schemes to address an information-theoretic interference suppression problem known as the Witsenhausen counterexample (Grover, Sahai, & Park 2010).

[23]Notably, Conway & Sloane (1999, page 12) state: "A related application that has not yet received much attention is the use of these packings for solving n-dimensional *search* or *approximation* problems"; this is exactly the problem focused on in Chapter 3.

Far and away the most elegant and practical application of *n*-dimensional sphere packings, however, is in the framing and understanding of *error correcting codes* (*ECCs*). *The reader is referred to Leech & Sloane (1971), Thompson (1983), Pless (1998), and Conway & Sloane (1999) for comprehensive reviews of this fascinating subject.* A brief overview of this field, drawn primarily from these four references, is given here to emphasize the existing practical relevance of *n*-dimensional sphere packings with $n > 3$; we aim to augment this list of practical applications significantly in Chapters 3 and 4 of the present work, based heavily on the various extensions of *n*-dimensional sphere packing theory developed in this work.

To proceed, define $\mathbf{F}_q$ as the set of symbols in a *finite field* (a.k.a. *Galois field*) of order q, where $q = p^a$ with p prime, and define $\mathbf{F}_q^n$ as the set of all vectors of order n with elements selected from $\mathbf{F}_q$. The cases of particular interest in this work are the *binary field* $\mathbf{F}_2 = \{0,1\}$, the *ternary field* $\mathbf{F}_3 = \{0,1,2\}$, and the *quaternary field* $\mathbf{F}_4 = \{0,1,\omega,\bar{\omega}\}$, where, as in §2.3.1, $\omega = (-1+\imath\sqrt{3})/2$ [note that $\omega^2 = \bar{\omega}$, $\bar{\omega}^2 = \omega$, and $\bar{\omega}\cdot\omega = 1$]. In a finite field $\mathbf{F}_q$, addition ($+$) and multiplication ($\cdot$) are closed (that is, they map to elements within the field) and satisfy the usual rules: they are associative, commutative, and distributive, there is a 0 element such that $a+0=a$, there is a 1 element such that $a\cdot 1 = a$, for each a there is an element $(-a)$ such that $a+(-a)=0$, and for each $a \neq 0$ there is an element a^{-1} such that $a\cdot a^{-1} = 1$. If q is itself prime (e.g., if $q=2$ or $q=3$), then standard integer addition and multiplication mod q forms a finite field. If not (e.g., if $q=4$), a bit more care is required in order to obtain closure within the finite field while respecting these necessary rules on addition and multiplication. For the cases considered in this section (specifically, $\mathbf{F}_2$, $\mathbf{F}_3$, and $\mathbf{F}_4$), addition and multiplication on $\mathbf{F}_q$ are thus defined as follows:

$\mathbf{F}_2$:

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

$\mathbf{F}_3$:

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\cdot$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

$\mathbf{F}_4$:

$+$	0	1	ω	$\bar{\omega}$
0	0	1	ω	$\bar{\omega}$
1	1	0	$\bar{\omega}$	ω
ω	ω	$\bar{\omega}$	0	1
$\bar{\omega}$	$\bar{\omega}$	ω	1	0

$\cdot$	0	1	ω	$\bar{\omega}$
0	0	0	0	0
1	0	1	ω	$\bar{\omega}$
ω	0	ω	$\bar{\omega}$	1
$\bar{\omega}$	0	$\bar{\omega}$	1	ω

An $[n,k]_q$ (or, when d is explicitly specified, $[n,k,d]_q$) *linear*[24] *q-ary*[25] *code* (*LC*) may now be defined as a *q-ary linear combination* (that is, a linear combination with coefficients selected from $\mathbf{F}_q$, with addition and multiplication defined elementwise on $\mathbf{F}_q$, as discussed above) of a set of $k < n$ independent *basis vectors* $\mathbf{v}^i_{[n,k]_q} \in \mathbf{F}_q^n$,

where the basis vectors are selected such that each of the q^k resulting *codewords* $\mathbf{w}^i_{[n,k]_q} \in \mathbf{F}_q^n$ (that is, each q-ary linear combination of the basis vectors) differs from each of the other codewords in at least d elements (referred to in this setting as the *Hamming distance*). We denote by $V_{[n,k]_q}$ (or $V_{[n,k,d]_q}$) the $n \times k$ *basis matrix* with the k basis vectors $\mathbf{v}^i_{[n,k]_q}$ as its columns, and by $W_{[n,k]_q}$ (or $W_{[n,k,d]_q}$) the $n \times q^k$ *codeword matrix* with the q^k codewords $\mathbf{w}^i_{[n,k]_q}$ as its columns. Further, without loss of generality, we construct $V_{[n,k]_q}$ and a companion $(n-k) \times n$ *parity check* matrix $H_{[n,k]_q}$ in the standard (a.k.a. *systematic*) form[26]

$$V_{[n,k]_q} = \begin{bmatrix} I_{k\times k} \\ P_{(n-k)\times k} \end{bmatrix} \quad \text{and} \quad H_{[n,k]_q} = \begin{bmatrix} -P_{(n-k)\times k} & I_{(n-k)\times(n-k)} \end{bmatrix}. \tag{2.16}$$

Note that $H_{[n,k]_q} V_{[n,k]_q} = 0$ (on $\mathbf{F}_q$)[27], which establishes that the basis vectors $\mathbf{v}^i_{[n,k]_q}$ so constructed [and, thus, all of the valid codewords $\mathbf{w}$ given by q-ary linear combination of these basis vectors] each satisfy the parity check equations, $H_{[n,k]_q}\mathbf{w} = 0$ (on $\mathbf{F}_q$), implied by the rows of $H_{[n,k]_q}$, as illustrated by the several examples given in this standard form in §2.5.1, §2.5.2, and §2.5.3. The first k symbols of an $[n,k]_q$ LC so defined are referred to as the *data symbols*[28], and the last $n-k$ symbols are referred to as the *parity symbols*.

[24] Nonlinear q-ary codes also appear in the literature, in which the valid codewords are *not* simply linear combinations of a set of basis vectors, but rather must be enumerated individually. Such codes, which are related to nonlattice packings, are in general more difficult to decode than LCs, and are not considered further here.

[25] This work focuses on the cases with $q = 2$ [termed a *linear binary code* (*LBC*)], $q = 3$ [termed a *linear ternary code* (*LTC*)], and $q = 4$ [termed a *linear quaternary code* (*LQC*)], in §2.5.1, §2.5.2, and §2.5.3 respectively. In cases with $q = 2$, which are the most common, we frequently write simply $[n,k]$ or $[n,k,d]$, dropping the q subscript.

[26] In the literature on this subject, it is more common to use a "generator matrix" G to describe the construction of linear codes. The "basis matrix" convention V used here is related simply to the corresponding generator matrix such that $V = G^T$; we find the basis matrix convention to be more natural in terms of its linear algebraic interpretation.

[27] The qualifiers "(on $\mathbf{F}_q$)" and "(mod q)" are used, as appropriate, to remind the reader that multiplication and addition in the equation indicated are performed elementwise on the finite field $\mathbf{F}_q$, as discussed above.

[28] The word "bit", a portmanteau word for "binary digit", is generally reserved for the case with $q = 2$; in the general case, we use the word "symbol" in its place.

The key to designing a "good" $[n,k]_q$ LC is to construct the *parity submatrix* $P_{(n-k)\times k}$ in (2.16) in such a way that the value of d in the resulting code (that is, the minimum Hamming distance between valid codewords) is maximized for given values of n, k, and q; significant effort was required for this construction during the development of some of the codes reviewed in §2.5.1, §2.5.2, and §2.5.3. Indeed, the problem of designing a good binary error correcting code is essentially a finite sphere packing problem on $\mathbf{F}_2$; thus the very close relationship between the design of error-correcting codes and the design of infinite sphere packings in $\mathbb{R}^n$, as discussed in §2.3.

For $q = p^a$ with p prime, *conjugation in* $\mathbf{F}_q$ (that is, for scalars $v \in \mathbf{F}_q$) is defined such that $\bar{v} = v^p$; *conjugation in* $\mathbf{F}_q^n$ (that is, for vectors $\mathbf{v} \in \mathbf{F}_q^n$), as well as for matrices formed with a number of such vectors as columns, is performed elementwise. The *dual code* of any $[n,k]_q$ LC is then the $[n,n-k]_q$ LC given by the union of all codewords $\mathbf{w} \in \mathbf{F}_q^n$ for which $\mathbf{w} \cdot \bar{\mathbf{v}} = 0$ (on $\mathbf{F}_q$) for all $\mathbf{v} \in \{\mathbf{v}^i_{[n,k]_q}$ for $i = 1,\ldots,k\}$ [cf. (2.2)]. The codeword and parity check matrices for this dual code may thus, when arranged in standard form, be written as

$$V_{[n,n-k]_q} = \begin{bmatrix} I_{(n-k)\times(n-k)} \\ -\bar{P}^T \end{bmatrix} \quad \text{and} \quad H_{[n,n-k]_q} = \begin{bmatrix} \bar{P}^T & I_{(n-k)\times(n-k)} \end{bmatrix}, \tag{2.17}$$

where $\bar{P}$ denotes conjugation in $\mathbf{F}_q$ of each element of the parity submatrix P of the original $[n,k]_q$ LC. Note that $\bar{P}^T$ is of order $k \times (n-k)$, and, of course, that $H_{[n,n-k]_q} V_{[n,n-k]_q} = 0$ (on $\mathbf{F}_q$). Note also that, for LBCs and LTCs, $\bar{P} = P$.

Graphically, the codewords of an $[n,k,d]_2$ LBC may be thought of as a carefully chosen subset of 2^k of the 2^n corners on a single n-dimensional unit hypercube,as illustrated for $n = 3$ in Figure 2.9, whereas an $[n,k,d]_3$ LTC may be thought of as a subset of 3^k of the 3^n gridpoints in a cluster of 2^n unit hypercubes in n-dimensions, as illustrated for $n = 3$ in Figure 2.10. For any q, d quantifies the minimum number of symbols which differ between any two codewords. It follows that:

- An LC with $d = 2$ is *single error detecting* (*SED*) [see, e.g., Figures 2.9a and 2.10a]. In this case, the sum (on $\mathbf{F}_q$) of the symbols in each valid transmitted codeword is zero, so if it is assumed that at most one symbol error occured and this sum is nonzero, then a symbol error in transmission occurred, whereas if it is zero, then a symbol error did not occur. However, if a symbol error in transmission occured, the received (invalid)

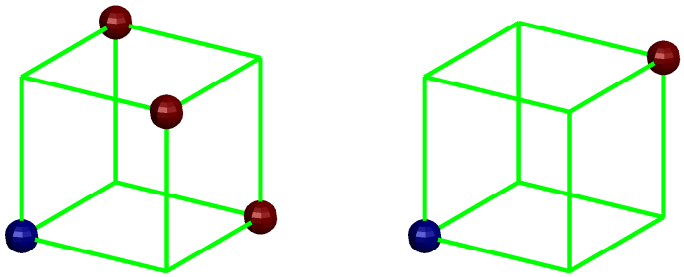

Figure 2.9: Valid codewords of (left) the (SED) $[3,2,2]_2$ LBC, and (right) its dual, the (perfect, SEC) $[3,1,3]_2$ LBC. In this case, d specifies the number of edges that separate any two valid codewords. The blue spheres denote the origin.

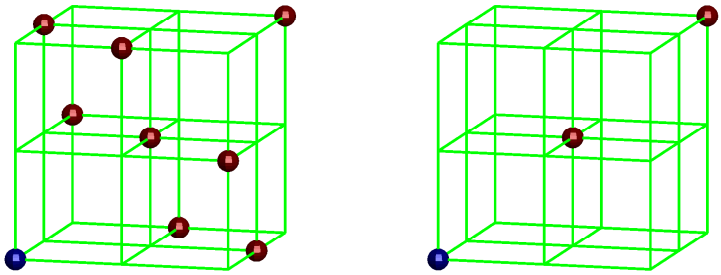

Figure 2.10: Valid codewords of (left) the (SED) $[3,2,2]_3$ LTC, and (right) its dual, the (SEC) $[3,1,3]_3$ LTC (cf. Figure 2.9).

codeword is generally equidistant from multiple valid codewords, so it is not possible to correct the symbol error. Two or more symbol errors generally cause the codeword to be misinterpreted.

- An LC with $d = 3$ is *single error correcting* (*SEC*) [see, e.g., Figures 2.9b and 2.10b]. In this case, if it is again assumed that at most one symbol error in transmission occured, then if the received codeword is not a valid codeword, there is only one valid codeword that is unit Hamming distance away, so the single symbol error may in fact be *corrected*. Again, 2 or more symbol errors generally cause the codeword to be misinterpreted.
- An LC with $d = 4$ is *single error correcting and double error detecting* (*SECDED*). In this case, if a single symbol error occurs, the received codeword will be unit Hamming distance away from a single valid codeword, and thus single symbol errors can be corrected. However, if two symbol errors occur, the received codeword is generally Hamming distance 2 away from multiple valid codewords, so double symbol errors

can be detected but *not* corrected. Now, 3 or more symbol errors generally cause the codewords to be misinterpreted.

- An LC with $d = 5$ is *double error correcting* (*DEC*), with 3 or more symbol errors generally causing misinterpretation.
- An LC with $d = 6$ is *double error correcting and triple error detecting* (*DECTED*), with 4 or more symbol errors generally causing misinterpretation.
- An LC with $d = 7$ is *triple error correcting* (*TEC*), with 4 or more symbol errors generally causing misinterpretation.
- An LC with $d = 8$ is *triple error correcting and quadruple error detecting* (*TECQED*), with 5 or symbol errors generally causing misinterpretation.
- An LC with $d = 9$ is *quadruple error correcting* (*QEC*), with 5 or more symbol errors generally causing misinterpretation.

The labels defined above are frequently used to quantify the error correction capability of an LC. Alternatively, if error correction is *not* attempted, then:

- An LC with $d = 2$ is single error detecting, with 2 or more symbol errors generally causing misinterpretation.
- An LC with $d = 3$ is double error detecting, with 3 or more symbol errors generally causing misinterpretation.
- An LC with $d = 4$ is triple error detecting, with 4 or more symbol errors generally causing misinterpretation.
- An LC with $d = 5$ is quadruple error detecting, with 5 or more symbol errors generally causing misinterpretation.

Error correcting algorithms are useful for a broad range of data transmission or data storage applications in which it is difficult or impossible to request that a corrupted codeword be retransmitted; algorithms which use such LCs for error detection only, on the other hand, may be used only when efficient handshaking is incorporated in a manner which makes it easy to request and resend any messages that might be corrupted during transmission.

An $[n,k,d]_q$ LC is called *perfect* if, for some integer $t > 0$, each possible n-dimensional q-ary codeword is of Hamming distance t or less from a single valid codeword (that is, if there are no "wasted" codewords which are Hamming distance $t+1$ or

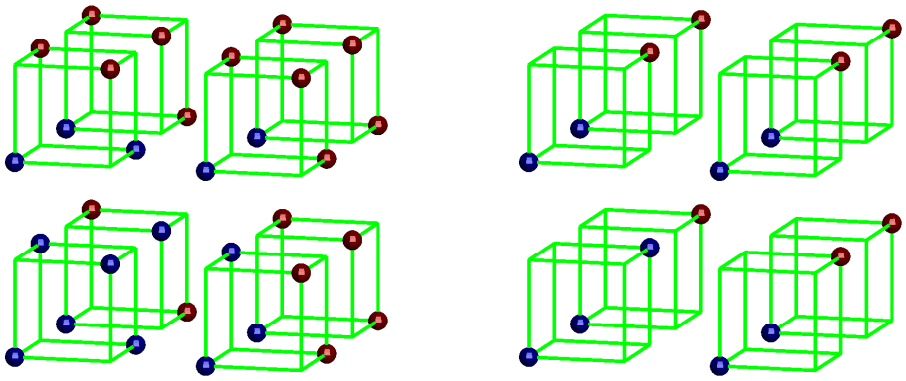

Figure 2.11: The lattice corresponding to a $[n,k,d]$ LBC is formed by repeating the unit hypercube pattern given by the LBC (see, e.g., Figure 2.9) as an infinite array with unit spacing. In the above example, we illustrate this extension for (left) the face-centered cubic (FCC) lattice generated by the $[3,2,2]$ LBC, $D_3 = \bigcup_{i=1}^{4} (\mathbf{w}^i_{[3,2,2]} + 2\mathbb{Z}^3)$, and (right) the body-centered cubic (BCC) lattice generated by the $[3,1,3]$ LBC, $D_3^* = \bigcup_{i=1}^{2} (\mathbf{w}^i_{[3,1,3]} + 2\mathbb{Z}^3)$. The blue spheres, taken together, form a *primitive cell* that, repeated as an infinite array with *zero* spacing, tile (that is, fill) the domain.

more from the valid codewords, and thus may not be corrected under the assumption that at most t symbol errors have occured); note that a perfect code has odd $d = 2t+1 > 1$. A remarkable proof by Tietäväinen (1973), which was based on related work by Van Lint, establishes that the *only* nontrivial perfect LCs are the $[(q^r-1)/(q-1),(q^r-1)/(q-1)-r,3]_q$ perfect q-ary Hamming codes and the $[23,12,7]_2$ and $[11,6,5]_3$ binary and ternary Golay codes, described further in §2.5.1 and §2.5.2.

An $[n,k,d]$ LC is called *quasi-perfect* if, for some integer $t > 1$, each possible n-dimensional q-ary codeword is either (a) of Hamming distance $t-1$ or less from a single valid codeword, and thus up to $t-1$ symbol errors may be corrected, or (b) of Hamming distance t from at least one valid codeword, and thus codewords with t symbol errors may be detected but not necessarily corrected (that is, again, if there are no "wasted" codewords which are Hamming distance $t+1$ or more from a valid codeword, and thus may not be reconciled under the assumption that at most t symbol errors have occured); note that a quasi-perfect code has even $d = 2t > 2$.

Note finally, as illustrated for $n = 3$ in Figure 2.11, that a real lattice corresponding to an $[n,k,d]_2$ LBC may often be constructed by forming a union of 2^k cosets:

$$\textit{Construction A}: \quad \bigcup_{i=1}^{2^k} (\mathbf{w}^i_{[n,k,d]_2} + 2\mathbb{Z}^n), \tag{2.18a}$$

where the *coset representatives* in this construction, $\mathbf{w}^i_{[n,k,d]_2}$ for $i = 1,\ldots,2^k$, are the codewords of the $[n,k,d]_2$ LBC under consideration and $(\mathbf{w} + 2\mathbb{Z}^n)$ denotes a $\mathbb{Z}^n$ lattice scaled by a factor of 2 with all nodal points shifted by the vector $\mathbf{w}$; thus, Construction A denotes the union of the nodal points in several such scaled and shifted $\mathbb{Z}^n$ lattices. An alternative real lattice may sometimes be constructed via:

$$\textit{Construction B}: \quad \bigcup_{i=1}^{2^k} (\mathbf{w}^i_{[n,k,d]_2} + 2J) \quad \text{where} \quad J = \left\{ \mathbf{x} \in \mathbb{Z}^n \,\middle|\, \left[\sum_{i=1}^{n} x_i \right] \in 2\mathbb{Z} \right\}, \tag{2.18b}$$

where $(2\mathbb{Z})$ denotes the even integers, and thus the last condition is sometimes written $\sum_{i=1}^{n} x_i = 0 \pmod 2$.

In an analogous fashion, a complex lattice corresponding to an $[n,k,d]_q$ LC may often be constructed by forming a union of q^k shifted and scaled n-dimensional $\mathscr{E}$ lattices $\mathbb{Z}[\omega]^n$ (see §2.3.1) such that

$$\textit{Construction } A^{\pi}_{\mathscr{E}}: \quad \bigcup_{i=1}^{q^k} (\mathbf{w}^i_{[n,k,d]_q} + \pi\,\mathbb{Z}[\omega]^n), \tag{2.19a}$$

where, in the sequel, the multiplicative factor π takes two possible values (2 and $\theta = \omega - \bar{\omega} = \imath\sqrt{3}$) and the coset representatives in this construction, $\mathbf{w}^i_{[n,k,d]_q}$ for $i = 1,\ldots,q^k$, are the codewords of the $[n,k,d]_q$ LC under consideration. An alternative complex lattice may sometimes be constructed via:

$$\textit{Construction } B^{\pi}_{\mathscr{E}}: \quad \bigcup_{i=1}^{q^k} (\mathbf{w}^i_{[n,k,d]_q} + \pi J) \quad \text{where} \quad J = \left\{ \mathbf{x} \in \mathbb{Z}[\omega]^n \,\middle|\, \left[\sum_{i=1}^{n} x_i \right] \in \pi\mathscr{E} \right\}, \tag{2.19b}$$

where $(\pi\mathscr{E})$ denotes the lattice of Eisenstein integers in the complex plane multiplied (that is, rotated and scaled) by the (possibly complex) factor π. Note the remarkable similarity in structure between the real constructions in (2.18a)-(2.18b) and the complex constructions in (2.19a)-(2.19b). Note also that real lattices corresponding to any of the complex lattices so constructed may easily be generated via (2.3).

2.5.1 Exemplary linear binary codes (LBCs)

We now summarize some of the families of LBCs available, describing each in the standard form (2.16):

- The simple[29] $[n, n-1, 2]$ *binary single parity check codes* are SED, and include $[2,1,2]$, $[3,2,2]$, $[4,3,2]$, $[5,4,2]$, etc. Using such a code, for each $(n-1)$ data bits to be transmitted, a parity bit is generated such that the sum (mod 2) of the data bits plus the parity bit is 0; when decoding, an error is flagged if this sum (mod 2) is 1. The $[3,2,2]$ code illustrated in Figure 2.9a is given by

$$V_{[3,2,2]} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad H_{[3,2,2]} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, \quad W_{[3,2,2]} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}. \tag{2.20}$$

  As seen for $n = 3$ in Figure 2.11a, via Construction A, the $[n, n-1, 2]$ binary single parity check code generates the D_n lattice (see §2.3.3), which for $n = 3$ is FCC.

- The dual of the binary single parity check codes are the simple $[n, 1, n]$ *binary repetition codes,* which include $[2,1,2]$ (SED), $[3,1,3]$ (SEC), $[4,1,4]$ (SECDED), $[5,1,5]$ (DEC), etc. (note that the $[2,1,2]$ code is self dual, and that the $[3,1,3]$ code is perfect). This family of codes just repeats any given data bit n times; when decoding, one simply needs to determine which of the two valid codewords that the received code is nearest to. The $[3,1,3]$ code illustrated in Figure 2.9b is given by

$$V_{[3,1,3]} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad H_{[3,1,3]} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \quad W_{[3,1,3]} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}. \tag{2.21}$$

  As seen for $n = 3$ in Figure 2.11b, via Construction A, the $[n, 1, n]$ binary repetition code generates the D_n^* lattice (see §2.3.3), which for $n = 3$ is BCC. Via Construction B, on the other hand, the $[8,1,8]$ binary repetition code generates the E_8 lattice (see

[29] As suggested in Footnote 25 on page 76, when $q = 2$, we may suppress the q subscript for notational clarity; we thus do this throughout §2.5.1.

§2.3.5). Note also that the $[3,2,2]$ binary single parity check code with each bit repeated m times leads to a $[3m,2,2m]$ code that may be rearranged into standard form and written as

$$V_{[3m,2,2m]} = \begin{bmatrix} I_{2\times 2} \\ P_{(3m-2)\times 2} \end{bmatrix}, \quad H_{[3m,2,2m]} = \begin{bmatrix} P_{(3m-2)\times 2} & I_{(3m-2)\times(3m-2)} \end{bmatrix}$$

$$P_{(3m-2)\times 2} = \begin{pmatrix} 1_{(m-1)\times 1} & 0_{(m-1)\times 1} \\ 0_{(m-1)\times 1} & 1_{(m-1)\times 1} \\ 1_{m\times 1} & 1_{m\times 1} \end{pmatrix}.$$

In particular, taking $m = 4$ and applying Construction B, the resulting $[12,2,8]$ code, which is TECQED, generates the $\Lambda_{12}^{\max}$ lattice (see §2.3.6).

- The $[2^m-1, 2^m-1-m, 3]$ *binary Hamming codes* are perfect and SEC, and include $[3,1,3]$, $[7,4,3]$, $[15,11,3]$, etc. For a given (2^m-1-m) data bits to be transmitted, each parity bit is generated such that the sum (mod 2) of a particular subset of the data bits plus that parity bit is 0; when decoding, the m parity bits may be used to determine not only whether or not a single bit error occured (which is true if one or more of these parity bits is nonzero), but if it did, *which* bit contains the error. To illustrate, the venerable $[7,4,3]$ code is given by

$$V_{[7,4,3]} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \quad H_{[7,4,3]} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$

$$W_{[7,4,3]} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{2.22}$$

Taking the matrix $H_{[7,4,3]}$ times any of the valid codewords $\mathbf{w}^i_{[7,4,3]}$ (listed in the columns of $W_{[7,4,3]}$) gives (mod 2) the zero vector, whereas taking the matrix $H_{[7,4,3]}$ times any invalid codeword gives (mod 2) a nonzero vector of order $n-k=3$, which may be interpreted as a nonzero 3-bit binary number, the integer corresponding to which we call the *syndrome* of the invalid codeword, denoted s. Conveniently, assuming no more than one bit error occurred, this number s may be used in a simple way in this particular class of codes to determine which bit must be flipped in the (invalid) codeword received to find the nearest valid codeword, thereby performing single error correction. To accomplish this, as easily verified by hand, denoting the k data bits of the codeword as d_i and the $n-k$ parity bits as p_i and reordering these bits as the vector $\{p_1, p_2, d_1, p_3, d_2, d_3, d_4\}$, it is the s'th element of this vector that must be flipped. Note also that, via Construction A, the $[7,4,3]$ binary Hamming code generates the E_7^* lattice (see §2.3.5).

- The dual of the binary Hamming codes are the $[2^k-1, k, 2^{k-1}]$ *binary simplex codes*, which include $[3,2,2]$ (SED), $[7,3,4]$ (SECDED), $[15,4,8]$ (TECQED), etc. These codes are remarkable geometrically, as their codewords form a simplex. We have

already illustrated the $[3,2,2]$ code in Figure 2.9a; the $[7,3,4]$ code is given by

$$V_{[7,3,4]} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \quad W_{[7,3,4]} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}, \tag{2.23}$$

$$H_{[7,3,4]} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Via Construction A, the $[7,3,4]$ binary simplex code generates the E_7 lattice (see §2.3.5). Via Construction B, on the other hand, the $[15,4,8]$ binary simplex code generates the Λ_{15} lattice (see §2.3.6).

- The $[2^m, 2^m-1-m, 4]$ *extended binary Hamming codes* are quasi-perfect and SECDED, and include $[4,1,4]$, $[8,4,4]$, $[16,11,4]$, etc. These codes are just binary Hamming codes with an additional overall parity bit, and thus, assuming no more than 2 bit errors have occured, may be decoded similarly. Defining the syndrome s as in the binary Hamming code (neglecting the overall parity bit), and defining p as the sum over all the bits (including the overall parity bit), there are zero bit errors if $s = p = 0$, there two bit errors (which may be detected but not uniquely corrected) if $s \neq 0$ and $p = 0$, and there is a single bit error if $p \neq 0$ (in which case, if $s = 0$, this error is in the overall parity bit, and, if $s \neq 0$, this error is in one of the other bits and may be corrected based on s just as in the binary Hamming code). To illustrate in standard form, the venerable $[8,4,4]$ code is given by[30]

[30]The standard form of the $[8,4,4]$ code shown here may be related to the perhaps more intuitive form of this code described previously in this paragraph by replacing the last row of the concomitant parity check matrix $P_{4\times 4}$ [see (2.16)] by the sum (mod 2) of all of the rows of P in the form given in (2.24). This results in a row with 1 in each of its elements, implying simply an overall parity check on the $[7,4,3]$ code in (2.22). These two forms are, of course, equivalent.

$$V_{[8,4,4]} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \quad H_{[8,4,4]} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$W_{[8,4,4]} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{2.24}$$

Via Construction A, the $[8,4,4]$ extended binary Hamming code again generates the E_8 lattice (see §2.3.5).

- The dual of the extended binary Hamming codes are the $[2^m, m+1, 2^{m-1}]$ *binary biorthogonal codes* (a.k.a. *Hadamard codes*), and include the $[4,3,2]$ (SED), $[8,4,4]$ (SECDED), $[16,5,8]$ (TECQED), $[32,6,16]$, etc. (note that the $[8,4,4]$ code is self dual). The $[32,6,16]$ code was used on the Mariner 9 spacecraft. These codes are distinguished by the characteristic that their codewords are mutually orthogonal [that is, $\mathbf{w}^i \cdot \mathbf{w}^j = 0 \pmod 2$ for $i \neq j$]. Note that the $[4,3,2]$ and $[8,4,4]$ codes have already been discussed above. Via Construction B, the $[16,5,8]$ binary biorthogonal code generates the Λ_{16} lattice (see §2.3.6).

- The $[n,(n+1)/2,d]$ *binary quadratic residue codes* are defined for all prime n for which there exists an integer $1 < x < n$ such that $x^2 = 2 \pmod n$ [equivalently, for all prime n of the form $n = 8m \pm 1$ where m is an integer], and include $[7,4,3]$ (SEC,

perfect, a.k.a. a binary Hamming code), $[17,9,5]$ (DEC), $[23,12,7]$ (TEC, perfect, a.k.a. the *binary Golay code*), $[31,16,7]$ (TEC), $[41,21,9]$ (QEC), $[47,24,11]$, etc. Adding an overall parity bit to these codes, the $[n+1,(n+1)/2,d+1]$ *extended binary quadratic residue codes* include $[8,4,4]$ (SECDED, quasi-perfect, self-dual, a.k.a. an extended binary Hamming code), $[18,9,6]$ (DECTED), $[24,12,8]$ (TECQED, quasi-perfect, self-dual, a.k.a. the *extended binary Golay code*), $[32,16,8]$ (TECQED), $[42,21,10]$, $[48,24,12]$, etc. The venerable $[24,12,8]$ extended binary Golay code, which was used by the Voyager 1 & 2 spacecraft, is given by

$$V_{[24,12,8]} = \begin{bmatrix} I_{12\times 12} \\ P_{12\times 12} \end{bmatrix}, \quad H_{[24,12,8]} = \begin{bmatrix} P_{12\times 12} & I_{12\times 12} \end{bmatrix},$$

$$P_{12\times 12} = \begin{pmatrix} 0&1&1&1&1&1&1&1&1&1&1&1 \\ 1&1&1&0&1&1&1&0&0&0&1&0 \\ 1&1&0&1&1&1&0&0&0&1&0&1 \\ 1&0&1&1&1&0&0&0&1&0&1&1 \\ 1&1&1&1&0&0&0&1&0&1&1&0 \\ 1&1&1&0&0&0&1&0&1&1&0&1 \\ 1&1&0&0&0&1&0&1&1&0&1&1 \\ 1&0&0&0&1&0&1&1&0&1&1&1 \\ 1&0&0&1&0&1&1&0&1&1&1&0 \\ 1&0&1&0&1&1&0&1&1&1&0&0 \\ 1&1&0&1&1&0&1&1&1&0&0&0 \\ 1&0&1&1&0&1&1&1&0&0&0&1 \end{pmatrix}; \tag{2.25}$$

the $[23,12,7]$ binary Golay code may be obtained from the matrix of basis vectors $V_{[23,12,7]}$ constructed as in (2.16) with $P_{11\times 12}$ given by any 11 rows of the matrix $P_{12\times 12}$ defined above. Via Construction B, the $[24,12,8]$ extended binary Golay code generates an intermediate lattice which may be joined with a translate of itself to construct the Λ_{24} lattice (see §2.3.6).

As illustrated above in the case of the binary Hamming code and the binary Golay code, a perfect code may be *extended* to a quasi-perfect code by adding an overall parity bit. A code obtained by the reverse of this process (that is, by removing a parity

bit to reduce the length of a code by one) is sometimes said to be *punctured*. In contrast, a code obtained by removing one or more data bits (in essence, simply setting the unneeded data bits to zero) is said to be *shortened*. A typical and common application is in error-correcting memory systems for computers, in which the data often comes naturally in blocks of 64 bits. Starting from the $[127,120,3]$ binary Hamming code, one may eliminate 56 data bits to create a shortened $[71,64,3]$ SEC code; alternatively, starting from the $[128,120,4]$ extended binary Hamming code, one may eliminate 56 data bits to create a shortened $[72,64,4]$ SECDED code. Many so-called ECC Memory chips are based on variants of such binary Hamming codes, which are extremely simple and fast to decode. Note also that, via Construction B, the $[21,9,8]$, $[20,8,8]$, and $[19,7,8]$ codes obtained by shortening the $[24,12,8]$ extended binary Golay code by 3, 4, or 5 data bits generate, respectively, the Λ_{21}, Λ_{20}, and Λ_{19} lattices (see §2.3.6).

Many of the binary codes introduced thus far fall within a larger family of codes collectively referred to as *Reed-Muller* codes, illustrated in Figure 2.12.

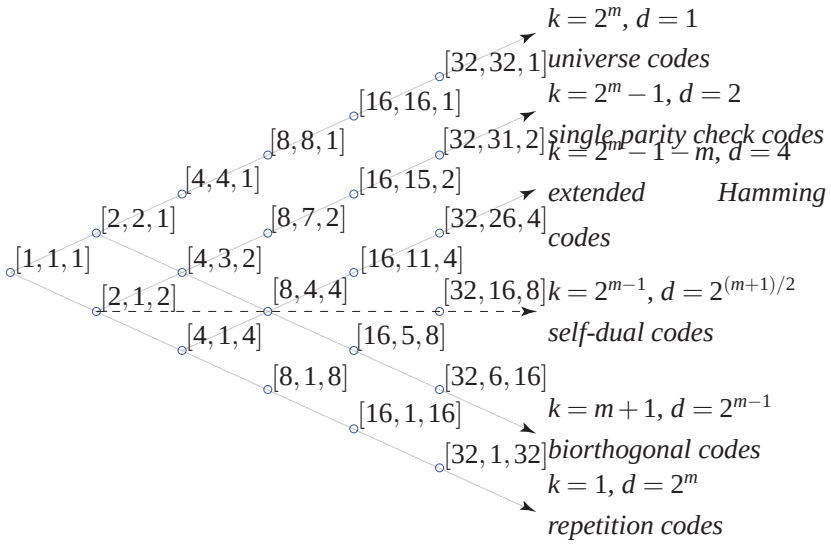

Figure 2.12: The family of $[2^m,k,d]$ Reed-Muller binary codes for $m = 0$ to 5.

2.5.2 Exemplary linear ternary codes (LTCs)

We now summarize some of the families of LTCs available, describing each in the standard form (2.16), noting that all have analogs in the binary setting:

- The $[n, n-1, 2]_3$ *ternary single parity check codes* are SED, and include $[2,1,2]_3$, $[3,2,2]_3$, $[4,3,2]_3$, etc. As illustrated for $n = 3$ in Figure 2.10a, the $[3,2,2]_3$ code is given by

$$V_{[3,2,2]_3} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 2 & 2 \end{pmatrix}, \quad H_{[3,2,2]_3} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, \quad W_{[3,2,2]_3} = \begin{pmatrix} 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 2 & 1 & 2 & 1 & 0 & 1 & 0 & 2 \end{pmatrix}. \tag{2.26}$$

  Via Construction $A^{\theta}_{\mathscr{C}}$, the $[3,2,2]_3$ ternary single parity check code generates the E_6^* lattice (see §2.3.5).

- The dual of the ternary single parity check codes are the $[n,1,n]_3$ *ternary repetition codes*, which include $[2,1,2]_3$ (SED), $[3,1,3]_3$ (SEC), $[4,1,4]_3$ (SECDED), etc. (note that the $[2,1,2]_3$ code is self dual). As illustrated for $n = 3$ in Figure 2.10b, the $[3,1,3]_3$ code is given by

$$V_{[3,1,3]_3} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad H_{[3,1,3]_3} = \begin{pmatrix} 2 & 1 & 0 \\ 2 & 0 & 1 \end{pmatrix}, \quad W_{[3,1,3]_3} = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 1 & 2 \end{pmatrix}. \tag{2.27}$$

  Via Construction $A^{\theta}_{\mathscr{C}}$, the $[3,1,3]_3$ ternary repetition code generates the E_6 lattice (see §2.3.5). Via Construction $B^{\theta}_{\mathscr{C}}$, on the other hand, the $[6,1,6]_3$ ternary repetition code generates the K_{12} lattice (see §2.3.6).

- The $[(3^m-1)/2, (3^m-1)/2-m, 3]_3$ *ternary Hamming codes* are perfect and SEC, and include $[4,2,3]_3$ (a.k.a. the *tetracode*), $[13,10,3]_3$, $[40,36,3]_3$, etc. To illustrate, the venerable $[4,2,3]_3$ tetracode is given by

$$V_{[4,2,3]_3} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 2 & 2 \\ 1 & 2 \end{pmatrix}, \quad W_{[4,2,3]_3} = \begin{pmatrix} 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 2 & 1 & 2 & 1 & 0 & 1 & 0 & 2 \\ 0 & 1 & 2 & 2 & 0 & 1 & 1 & 2 & 0 \end{pmatrix}, \qquad H_{[4,2,3]_3} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix}. \tag{2.28}$$

Via Construction $A^{\theta}_{\mathscr{E}}$, the $[4,2,3]_3$ tetracode again generates the E_8 lattice (see §2.3.5).

- The dual of the ternary Hamming codes are the $[(3^k-1)/2, k, 3^{k-1}]_3$ *ternary simplex codes*, which include $[4,2,3]_3$ (SEC), $[13,3,9]_3$ (QEC), $[40,4,27]_3$, etc. (note that the $[4,2,3]_3$ code is self dual). These codes are remarkable geometrically, as their codewords are all equidistant from one another.

- The $[n,(n+1)/2,d]_3$ *ternary quadratic residue codes* are defined for all prime n for which there exists an integer $1 < x < n$ such that $x^2 = 3 \pmod{n}$, and include $[11,6,5]_3$ (DEC, perfect, a.k.a. the *ternary Golay code*), $[13,7,5]_3$ (DEC), $[23,12,8]_3$ (TECQED), $[37,19,10]_3$, $[47,24,14]_3$, etc. Adding an overall parity bit to these codes, the $[n+1,(n+1)/2,d+1]_3$ *extended ternary quadratic residue codes* include $[12,6,6]_3$ (DECTED, quasi-perfect, self-dual, a.k.a. *the extended ternary Golay code*), $[14,7,6]_3$ (DECTED), $[24,12,9]_3$ (QEC), $[38,19,11]_3$, $[48,24,15]_3$, etc. The venerable $[12,6,6]_3$ extended ternary Golay code is given by

$$V_{[12,6,6]_3} = \begin{bmatrix} I_{6\times 6} \\ P_{6\times 6} \end{bmatrix}, \quad H_{[12,6,6]_3} = \begin{bmatrix} -P_{6\times 6} & I_{6\times 6} \end{bmatrix}, \quad P_{6\times 6} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 2 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 2 & 1 & 0 & 1 & 2 \\ 1 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 0 \end{pmatrix}; \tag{2.29}$$

the (punctured) $[11,6,5]_3$ code may be obtained from the matrix of basis vectors $V_{[11,6,5]}$ constructed as in (2.16) with $P_{5\times 6}$ given by any 5 rows of the matrix $P_{6\times 6}$

defined above. Via Construction $B^{\theta}_{\mathscr{C}}$, the $[12,6,6]_3$ extended ternary Golay code generates an intermediate lattice which may be joined with two translates of itself to generate the Λ_{24} lattice (see §2.3.6).

2.5.3 Exemplary linear quaternary codes (LQCs)

We now summarize some of the families of LQCs available, describing each in the standard form (2.16):

- The $[n,n-1,2]_4$ *quaternary single parity check codes* are SED, and include $[2,1,2]_4$, $[3,2,2]_4$, $[4,3,2]_4$, etc. The $[3,2,2]_4$ code is given by

$$V_{[3,2,2]_4} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad H_{[3,2,2]_4} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix},$$
$$W_{[3,2,2]_4} = \begin{pmatrix} 0 & 1 & \omega & \bar{\omega} & 0 & 1 & \omega & \bar{\omega} & 0 & 1 & \omega & \bar{\omega} & 0 & 1 & \omega & \bar{\omega} \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \omega & \omega & \omega & \omega & \bar{\omega} & \bar{\omega} & \bar{\omega} & \bar{\omega} \\ 0 & 1 & \omega & \bar{\omega} & 1 & 0 & \bar{\omega} & \omega & \omega & \bar{\omega} & 0 & 1 & \bar{\omega} & \omega & 1 & 0 \end{pmatrix}. \tag{2.30}$$

- The dual of the quaternary single parity check codes are the $[n,1,n]_4$ *quaternary repetition codes*, which include $[2,1,2]_4$ (SED), $[3,1,3]_4$ (SEC), $[4,1,4]_4$ (SECDED), etc. (note that the $[2,1,2]_4$ code is self dual). The $[3,1,3]_4$ code is given by

$$V_{[3,1,3]_4} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad H_{[3,1,3]_4} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \quad W_{[3,1,3]_4} = \begin{pmatrix} 0 & 1 & \omega & \bar{\omega} \\ 0 & 1 & \omega & \bar{\omega} \\ 0 & 1 & \omega & \bar{\omega} \end{pmatrix}. \tag{2.31}$$

- The $[(4^m-1)/3,(4^m-1)/3-m,3]_4$ *quaternary Hamming codes* are perfect and SEC, and include $[5,3,3]_4$, $[21,18,3]_4$, $[85,81,3]_4$, etc. The $[(4^m-1)/3+1,(4^m-1)/3-m,4]_4$ *extended quaternary Hamming codes* are quasi-perfect and SECDED, and include $[6,3,4]_4$ (a.k.a. the *hexacode*), $[22,18,4]_4$, $[86,81,4]_4$, etc. To illustrate, the venerable $[6,3,4]_4$ hexacode, which has $4^3 = 64$ valid codewords, is given by

$$V_{[6,3,4]_4} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & \omega & \omega \\ \omega & 1 & \omega \\ \omega & \omega & 1 \end{pmatrix}, \quad H_{[6,3,4]_4} = \begin{pmatrix} 1 & \omega & \omega & 1 & 0 & 0 \\ \omega & 1 & \omega & 0 & 1 & 0 \\ \omega & \omega & 1 & 0 & 0 & 1 \end{pmatrix}, \tag{2.32}$$

Via Construction $A^2_{\mathscr{E}}$, the $[6,3,4]_4$ hexacode generates the K_{12} lattice. The $[6,3,4]_4$ hexacode also forms the basis for the Miracle Octad Generator (MOG) construction of the Λ_{24} lattice, as discussed in Chapter 11 of Conway & Sloane (1999).

- The dual of the quaternary Hamming codes are the $[(4^k-1)/3, k, 4^{k-1}]_4$ *quaternary simplex codes*, which include $[5,2,4]_4$ (SECDED), $[21,3,16]_4$, $[85,4,64]_4$, etc. These codes are remarkable geometrically, as their codewords are all equidistant from one another.

We note finally that the simple low-dimensional LBC, LTC, and LQC constructions given above are now supplanted by the more complex *low-density parity-check* (*LDPC*, a.k.a. *Gallager*) codes and *turbo* codes for high performance coding applications such as 10GBase-T ethernet and deep space communication. For more information on these codes, the reader is referred to Gallager (1963), Berrou *et al.* (1993), and Moon (2005).

2.6 Quantization (that is, moving onto a Lattice)

For convenience, we now review briefly some methods (adapted from Chapter 20 of Conway & Sloane 1999) for quantization from an arbitrary point $\mathbf{x}$ in $\mathbb{R}^n$ onto a point $\tilde{\mathbf{x}}$ on the discrete lattice, which is defined via integer linear combination of the columns of the corresponding basis matrix B. The solution to this problem is lattice specific, and thus is treated lattice by lattice in the subsections below. Note that we neglect the problem of scaling of the lattices in this discussion, which is trivial to implement in code. For brevity, our review below focuses on quantization to the root lattices up

to dimension $n = 8$. Quantization to the lattices Λ_n and K_n in dimension $9 \leq n \leq 24$ is a more difficult problem, and is the subject of numerous papers in the area of coding theory (see §2.5) in the last 20 years, where many efficient algorithms for such problems have been proposed and tuned.

Quantization to $\mathbb{Z}^n$

Quantize to $\mathbb{Z}^n$ simply by rounding each element of $\mathbf{x}$ to the nearest integer.

Quantization to D_n

Quantize to D_n by rounding $\mathbf{x}$ two different ways:

- Round each element of $\mathbf{x}$ to the nearest integer, and call the result $\hat{\mathbf{x}}$.
- Round each element of $\mathbf{x}$ to the nearest integer *except* that element of $\mathbf{x}$ which is furthest from an integer, and round that element the wrong way (that is, round it down instead of up, or up instead of down); call the result $\hat{\hat{\mathbf{x}}}$.

Compute the sum s of the individual elements of $\hat{\mathbf{x}}$; the desired quantiziation is $\tilde{\mathbf{x}} = \hat{\mathbf{x}}$ if is s is even, and $\tilde{\mathbf{x}} = \hat{\hat{\mathbf{x}}}$ if s is odd.

Quantization to A_n

The A_n lattice is defined in an n-dimensional subspace C of $\mathsf{Y} = \mathbb{R}^{n+1}$. The subspace C is spanned by the n columns of the corresponding basis matrix B_{A_n}, and the orthogonal complement of C is spanned by the vector $\mathbf{n}_{A_n}$. Thus, the nearest point in the subspace, $\mathbf{y}_\mathsf{C} \in \mathsf{C}$, to any given point $\mathbf{y} \in \mathsf{Y}$ is given by

$$\mathbf{y}_\mathsf{C} = \mathbf{y} - (\mathbf{y}, \mathbf{n}_{A_n}) \cdot \mathbf{n}_{A_n}.$$

An orthogonal basis $\hat{B}_{A_n}$ of C may easily be determined from B_{A_n} via Gram Schmidt orthogonalization. With this orthogonal basis, the vectors $\mathbf{x} \in \mathbb{R}^n$ comprising the A_n lattice may be related to the corresponding vectors $\mathbf{y}_\mathsf{C} \in \mathsf{C} \subset \mathsf{Y}$ (that is, on an n-dimensional subspace of $\mathbb{R}^{n+1}$) via the equation

$$\mathbf{y}_\mathsf{C} = \hat{B}_{A_n}\mathbf{x}. \tag{2.33a}$$

Thus, starting from some point $\mathbf{x} \in \mathbb{R}^n$ but not yet quantized onto the lattice, we can easily determine the corresponding $(n+1)$-dimensional vector $\mathbf{y}_\mathsf{C}$ which lies within the n-dimensional subspace C of $\mathbb{R}^{n+1}$ via (2.33a). Given this value of $\mathbf{y}_\mathsf{C} \in \mathsf{C}$, we now need to quantize onto the lattice. We may accomplish this with the following simple steps:

- Round each component of $\mathbf{y}_\mathsf{C}$ to the nearest integer, and call the result $\hat{\mathbf{y}}$. Define the deficiency $\Delta = \sum_i \hat{y}_i$, which quantifies the orthogonal distance of the point $\hat{\mathbf{y}}$ from the subspace C.
- If $\Delta = 0$, then $\tilde{\mathbf{y}} = \hat{\mathbf{y}}$. If not, define $\mathbf{d} = \mathbf{y}_\mathsf{C} - \hat{\mathbf{y}}$, and distribute the integers $0, \ldots, n$ among the indices $i_0, \ldots, i_n$ such that

$$-1/2 \leq d(\hat{y}_{i_0}) \leq d(\hat{y}_{i_1}) \leq \ldots \leq d(\hat{y}_{i_n}) \leq 1/2.$$

  If $\Delta > 0$, then nudge $\hat{\mathbf{y}}$ back onto the C subspace by defining $\tilde{y}_{i_k} = \begin{cases} \hat{y}_{i_k} - 1 & k < \Delta, \\ \hat{y}_{i_k} & \text{otherwise.} \end{cases}$

  If $\Delta < 0$, then nudge $\hat{\mathbf{y}}$ back onto the C subspace by defining $\tilde{y}_{i_k} = \begin{cases} \hat{y}_{i_k} + 1 & k > n + \Delta, \\ \hat{y}_{i_k} & \text{otherwise.} \end{cases}$

Back in n-dimensional parameter space, the quantized value $\tilde{\mathbf{y}} \in \mathsf{C}$ corresponds to

$$\tilde{\mathbf{x}} = \hat{B}_{A_n}^T \tilde{\mathbf{y}}. \tag{2.33b}$$

Quantization to the union of cosets

The dual lattices D_n^* and A_n^*, the triangular lattice A_2, and the packing D_n^+ (including the lattice $E_8 \cong E_8^* \cong D_8^+$) are described via the union of simple, real cosets in (2.5a), (2.8a), (2.7c), and (2.6), respectively. The lattices E_7 and E_7^* may be built via the union of simple, real cosets via Construction A [see (2.18a)], with coset representatives $\mathbf{w}^i_{[n,k,d]}$ given in (2.22) and (2.23) respectively. To quantize a lattice described in such a manner (as a union of simple cosets), one may quantize to each coset independently, then select from these individual quantizations that lattice point which is nearest to the original point $\mathbf{x}$.

The lattices E_6 and E_6^* may be built via the union of complex cosets [which are scaled and shifted complex $\mathscr{E}$ lattices $\mathbb{Z}[\omega]^3$] via Construction $A_\mathscr{E}^\pi$ [see (2.19a)], with

coset representatives $\mathbf{w}^i_{[n,k,d]}$ given in (2.27) and (2.26) respectively. Following Conway & Sloane (1984), to discretize a point $\mathbf{x}$ to coset i in these cases:

- Determine the complex vector $\mathbf{z} \in \mathbb{C}^3$ corresponding to $\mathbf{x} \in \mathbb{R}^6$. Shift and scale such that $\hat{\mathbf{z}} = (\mathbf{z} - \mathbf{a}_i)/\theta$.
- Determine the real vector $\hat{\mathbf{x}} \in \mathbb{R}^6$ corresponding to $\hat{\mathbf{z}} \in \mathbb{C}^3$. Quantize the first, second, and third pairs of elements of $\hat{\mathbf{x}}$ to the real triangular A_2 lattice to create the quantized vector $\hat{\hat{\mathbf{x}}}$.
- Determine the complex vector $\hat{\hat{\mathbf{z}}} \in \mathbb{C}^3$ corresponding to $\hat{\hat{\mathbf{x}}} \in \mathbb{R}^6$. Unscale and unshift such that $\tilde{\mathbf{z}} = \theta\hat{\hat{\mathbf{z}}} + \mathbf{a}_i$.
- Determine the real vector $\tilde{\mathbf{x}} \in \mathbb{R}^6$ corresponding to $\tilde{\mathbf{z}} \in \mathbb{C}^3$.

2.7 Conclusions

In short, §2.3 of this work is about generalizing to higher dimensions the familiar triangular, BCC, and FCC lattices, which are dense alternatives to the cubic lattice with reduced nonuniformity, whereas §2.4 of this work is about generalizing to higher dimensions a few (specifically, the most regular) of the many familiar nets arising in biology and crystallography, such as the honeycomb, diamond, and quartz graphs, which are rare alternatives to the cubic lattice with reduced coordination number. The primary successful application of n-dimensional sphere packing theory to date is in coding theory, as reviewed in §2.5. A working understanding of this material is essential for the new practical applications of lattice theory to be studied in Chapters 3 and 4 of this work, both of which leverage heavily the foundational material discussed here.

2.8 Acknowledgements

Chapter 2, in full, is a reprint of the material as it has been submitted to SIAM Review 2011. Bewley, T., Belitz, P., and Cessna, J., 2011. The dissertation author was the secondary investigator and author of this paper.

Chapter 3

New Horizons in Sphere-Packing Theory, Part II: Lattice-Based Derivative-free Optimization via Global Surrogates

Derivative-free algorithms are frequently required for the optimization of non-smooth scalar functions in n dimensions resulting, for example, from physical experiments or from the statistical averaging of numerical simulations of chaotic systems such as turbulent flows. The core idea of all efficient algorithms for problems of this type is to keep function evaluations far apart until convergence is approached. *Generalized pattern search* (*GPS*) algorithms, a modern class of methods particularly well suited to such problems, accomplish this by coordinating the search with an underlying grid which is refined, and coarsened, as appropriate. One of the most efficient subclasses of GPS algorithms, known as the *surrogate management framework* (*SMF*; see Booker *et al.* 1999), alternates between an exploratory *search* over an interpolating function which summarizes the trends exhibited by existing function evaluations, and an exhaustive *poll* which checks the function on neighboring points to confirm or confute the local optimality of any given *candidate minimum point* (*CMP*) on the underlying grid. The original SMF algorithm implemented a GPS step on an underlying Cartesian grid, augmented with a

Kriging-based surrogate search. Rather than using the n-dimensional Cartesian grid (the typical choice), the present work introduces for this purpose the use of lattices derived from n-dimensional sphere packings. As reviewed and analyzed extensively in Part I of this series, such lattices are significantly more uniform and have many more nearest neighbors than their Cartesian counterparts. Both of these facts make them far better suited for coordinating GPS algorithms, as demonstrated here in a variety of numerical tests.

3.1 Introduction

The minimization of computationally expensive, high-dimensional functions is often most efficiently performed via gradient-based optimization algorithms such as nonlinear conjugate gradients and L-BFGS-B. In complex systems for which an accurate computer model is available, the gradient required by such algorithms may often be found via adjoint analysis. However, when the function in question is not sufficiently smooth to leverage gradient information effectively during its optimization (see, e.g., Figure 3.1), a derivative-free approach is necessary. Such a scenario is evident, for example, when optimizing a finite-time-average approximation of an infinite-time-average statistic of a chaotic system such as a turbulent flow. Such an approximation may be determined via simulation or experiment. The truncation of the averaging window used to determine this approximation renders derivative-based optimization strategies ill suited, as the truncation error, though small, is effectively decorrelated from one flow simulation/experiment to the next. This effective decorrelation of the truncation error is reflected by the exponential growth, over the entire finite time horizon considered, of the adjoint field related to the optimization problem of interest in the simulation-based setting.

Due to the often significant expense associated with performing repeated function evaluations (in the above example, turbulent flow simulations or experiments), a derivative-free optimization algorithm which converges to within an accurate tolerance of the global minimum of a nonconvex function of interest with a minimum number of function evaluations is desired. It is noted that, in the general case, proof of convergence

of an optimization algorithm to a global minimum is possible only when, in the limit of a large number of function evaluations N, the function evaluations become dense in the feasible region of parameter space (Torn & Zilinskas, 1987). Though the algorithm developed in the present work, when implemented properly, satisfies this condition, so do far inferior approaches, such as a rather unintelligent algorithm which we call Exhaustive Sampling (ES), which simply covers the feasible parameter space with a grid, evaluates the function at *every* gridpoint, refines the grid by a factor of two, and repeats until terminated. Thus, a guarantee of global convergence is not sufficient to establish the *efficiency* of an optimization algorithm. If function evaluations are relatively expensive, and thus only a relatively small number of function evaluations can ultimately be afforded, effective heuristics for rapid convergence are perhaps even more important than rigorous proofs of the behavior of the optimization algorithm in the limit of large N, a limit that might actually be argued to be of limited relevance when function evaluations are expensive. Given that such algorithms are often used in applications in which only a few hundred function evaluations can be afforded, careful attention to such heuristics forms an important foundation for the present study.

One of the earliest derivative-free optimization approaches to appear in the literature is the *downhill simplex method* (see Spendley, Hext, & Himsworth 1962 and Nelder & Mead 1965). The downhill simplex method is inherently based on an iterative, amoeba-like evolution (moving one point at a time) of a set of $n+1$ points in n dimensions towards the minimum of a (possibly nonsmooth) function. A large body of literature appeared after the original introduction of this method, much of which was aimed at heuristic strategies designed to keep the evolving simplex as regular as possible as the iteration proceeds, while expanding or contracting as appropriate. The grid-based methods considered in the present work are fundamentally different, so we will not dwell on such grid-free methods in this introduction. However, it is worth noting the inherent dependence on the regularity an evolving *simplex* (that is, on an n-dimensional polytope with $n+1$ vertices) in this classical method, and an analogous focus in the present work on the identification (see §3.2.2) and characterization (see §3.2.1 and 3.2.4) of a maximally-uniform simplex (referred to in the present work as a *minimum positive basis*) around the best point encountered thus far as the iteration proceeds, referred to in

the present work as a *candidate minimum point*. The role of the simplex in both cases is essentially identical: to identify the best direction to move next using a minimum number of new function evaluations.

If, for the moment, we give up on the goal of global convergence, the perhaps simplest grid-based derivative-free optimization algorithm, which we call Successive Polling (SP), proceeds as follows:

- Start with a coarse grid and evaluate the function at some starting point on this grid, identified as the first candidate minimum point (CMP).
- Then, poll (that is, evaluate) the function values on gridpoints which neighbor the CMP in parameter space, at a sufficient number of gridpoints to *positively span*[1] the feasible neighborhood of the CMP [this step ensures convergence, as discussed further in Torczon 1997, Booker *et al.* 1999, and Coope & Price 2001]. When polling:

(a) If any poll point is found to have a function value less than that of the CMP, immediately consider this new point the new CMP and terminate the present poll step.

(b) If no poll points are found to have function values less than that of the CMP, refine the grid by a factor of two.

- Initiate a new poll step, either (a) around the new CMP or (b) around the old CMP on the refined grid, and repeat until terminated.

Though the basic SP algorithm described above, on its own, is not very efficient, there are a variety of effective techniques for accelerating it. All grid-based schemes which effectively build on this basic SP idea are classified as GPS algorithms.

The most efficient subclass of GPS algorithms, known as the Surrogate Management Framework (SMF; see Booker *et al*., 1999), leverages inexpensive interpolating "surrogate" functions (often, Kriging interpolations are used) to summarize the trends of the existing function evaluations, and to provide suggested new regions of parameter space in which to perform one or more additional function evaluation(s) between each poll step. SMF algorithms thus alternate beween two steps:

(i) *Search* over the inexpensive interpolating function to identify, based on the existing function evaluations, the most promising gridpoint at which to perform a new func-

[1]That is, such that any feasible point in the neighborhood of the CMP can be reached via a *linear combination with non-negative coefficients* of the vectors from the CMP to the poll points.

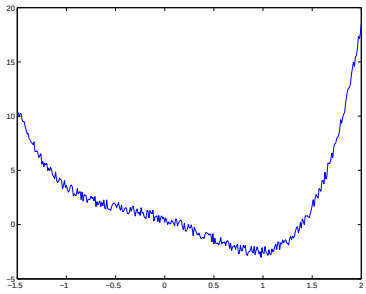

Figure 3.1: Prototypical nonsmooth optimization problem for which local gradient information is ill suited to accelerate the optimization algorithm.

tion evaluation. Perform a function evaluation at this point, update the interpolating function, and repeat. The search step may be terminated either when it returns a gridpoint at which the function has already been evaluated, or when the function, once evaluated, has a value greater than that of the CMP.

(ii) *Poll* the neighborhood of the new CMP identified by the search algorithm, following rules (a) and (b) above.

There is substantial flexibility during the search step described above. An effective search is essential for an efficient SMF algorithm. In the case that the search behaves poorly and fails to return improved function values, the SMF algorithm essentially reduces to the SP algorithm. If, however, the surrogate-based search is effective, the SMF algorithm will converge to a minimum far faster than a simple SP-based minimization. As the search and poll steps are essentially independent of each other, we will discuss them each in turn in the sections that follow, then present how we have combined them.

Note that if the search produces a new CMP which is several gridpoints away from the previous function evaluations, which occasionally happens when exploring functions with multiple minima, the grid may be *coarsened* appropriately in order to explore the vicinity of this new CMP efficiently (that is, with a coarse grid first, then refined as necessary). Note also that the interpolating surrogate function of the SMF may be used to *order* the function evaluations of the poll step, such that those poll points which are most likely to have a function value lower than that of the CMP are evaluated first. By so doing, the poll steps will, on average, terminate sooner, and the computational cost of the overall algorithm may be reduced further.

Table 3.1: Characteristics of select distinct lattices in dimensions 2, 3, and 8, ordered from dense to rare (for a more complete characterization, see Tables 2 and 3 of Chapter 2). Listed (see Chapter 2) are the packing density, Δ, covering thickness, Θ, mean squared quantization error per dimension, G, and kissing number, τ. Note that $\mathbb{Z}^n$ is significantly outperformed in every standard metric in every dimension $n > 1$ by the available alternatives.

n	lattice	name	Δ	Θ	G	τ
2	A_2	hexagonal	0.90690	1.2092	0.080188	6
	$\mathbb{Z}^2$	square	0.78540	1.5708	0.083333	4
3	A_3	face-centered cubic (FCC)	0.74048	2.0944	0.078745	12
	A_3^*	body-centered cubic (BCC)	0.68017	1.4635	0.078543	8
	$\mathbb{Z}^3$	cubic	0.52360	2.7207	0.083333	6
8	E_8	Gosset	0.25367	4.0587	0.071682	240
	D_8		0.12683	32.470	0.075914	112
	A_8	zero-sum	0.08456	32.993	0.077391	72
	D_8^*		0.03171	8.1174	0.074735	16
	A_8^*		0.02969	3.6658	0.075972	18
	$\mathbb{Z}^8$	Cartesian	0.01585	64.939	0.083333	16

To the best of our knowledge, all previous GPS and SMF implementations have been coordinated using Cartesian grids. A primary goal of the present work is to demonstrate convincingly that significant performance gains may be realized simply by eschewing this dominant Cartesian paradigm. Like in the game of checkers (contrast "American" checkers with "Chinese" checkers), Cartesian grids are not the only choice for discretizing parameter space. Other structured choices arising from n-dimensional sphere packing theory (see Tables 2.1 and 2.2, and further discussion in Chapter 2) are significantly more uniform and have many more nearest neighbors, especially as the dimension of the problem in question is increased; both of these properties suit these alternative lattices well for coordinating grid-based optimization algorithms.

Table 3.2: The densest, most uniform lattices available in several dimensions, and two factors quantifying the degree to which these lattices are better than the corresponding Cartesian grid in the same dimension; f_Δ denotes the factor of improvement in the packing density, an indication of the uniformity of the lattice, and f_τ denotes the factor of improvement in the kissing number, an indication of the flexibility available in selecting a positive basis from the nearest neighbors on the lattice. Note that the improvements becoming especially pronounced as the dimension n is increased.

	A_2	A_3	D_4	D_5	E_6	E_7	E_8	K_{12}	Λ_{16}	Λ_{24}
f_Δ	1.155	1.414	2	2.83	4.62	8	16	152	4096	$1.68e^7$
f_τ	1.5	2	3	4	6	9	15	31.5	135	4095

The definitive comprehensive reference on the subject of n-dimensional sphere packing theory is Conway & Sloane (1998)[2]. Chapter 2 of this study contains a concise summary of this involved subject, describing essentially everything one needs to know about lattices up to dimension $n = 24$ in order to use them effectively in practical engineering applications. For simplicity, the present investigation focuses on the use of just two such lattices, the zero-sum lattice A_n, which is an n-dimensional analog of the 2-dimensional hexagonal lattice and the 3-dimensional face-centered-cubic lattice, and the Gosset lattice E_8, which is an 8-dimensional analog of the 3-dimensional diamond packing, and is especially uniform. Both are described completely in Chapter 2; for brevity, this review will not be repeated here.

3.2 Extending lattice theory for derivative-free optimization

To extend the lattice theory described in Chapter 2 of this study in order to coordinate a derivative-free optimization, a few additional component algorithms are needed,

[2]In fact, as pointed out in Chapter 2, Conway & Sloane (1998, p. 12) state: "A related application that has not yet received much attention is the use of these packings for solving n-dimensional search or approximation problems"; this is exactly the focus of the present work.

which are now described. We begin with a short historical retrospective.

Thomson (1904), in his study of the structure of the atom, is credited with being the first to address the problem[3]: "Where should k inimical dictators settle on a planet in order to be as far away from each other as possible?" This question extends naturally to n-dimensional planets, and has received significant attention in the years since Thomson's original paper. The question is readily answered numerically by assigning an identical "charge" to each of n identical "particles", restricting particle motion to the surface of the sphere, and iteratively moving each particle (with some damping applied) in the direction of the force caused by the other particles (projected onto the sphere) until all particles come to equilibrium. The precise solution reached is a function of the distance metric and power law used when computing the force between any two particles; in the electrostatic setting, Thomson used the Euclidian distance between the particles, and a force which is proportional to the inverse square of this distance. The setting based on other distance measures (e.g., measured along the surface of the sphere instead of along a straight line) and other power laws are referred to as generalized Thomson problems; in particular, the case based on the p'th power in the limit that $p \to \infty$ (that is, the max value) was studied in Tammes (1930), in his study of the boundaries of pollen grains.

We now generalize this classical question in two ways, and introduce a new metric to characterize the solution found:

- First, the locations where the particles are allowed to settle are restricted to a discrete set of points on a sphere, which are specified as the nearest-neighbor lattice points to the CMP.
- Next, we allow some the particles' locations on the sphere to be specified (that is, fixed) in advance, and only move the remaining (free) particles to arrive at the best solution possible.
- Finally, the new metric we introduce is a check of whether or not the distribution produced by numerical solution of the resulting "discrete Thomson problem" forms a *positive basis* of the feasible neighborhood of the CMP; that is, in the case with no

[3]This curious problem, articulated by Meschkowski (1960) in terms of inimical dictators (see also L. Fejes Toth 1971), assumes that all locations on the planet's surface are equally desirable, and that the inimical dictators all cooperate.

active constraints (cf. §4.4.4), whether or not all points on the unit sphere around the CMP can be reached via a linear combination *with non-negative coefficients* of the vectors from the CMP to the optimized particle locations.

After developing a method to test for a positive basis, the remainder of this section develops three efficient algorithms to iterate on this "discrete Thomson problem" until a positive basis is found. To accomplish this, these algorithms first solve the discrete Thomson problem numerically for $n+m$ particles where $m=1$. If the optimization algorithm succeeds in producing a positive basis, the algorithm exits; otherwise, m is increased by one and the process repeated until a positive basis is determined. The resulting algoroithm is leveraged heavily during the poll step of the lattice-based SMF algorithm developed later in this work.

3.2.1 Testing for a positive basis

Given a subset of the nearest-neighbor lattice points, we will at times need an efficient test to determine whether or not the vectors to these points from the CMP form a positive basis of the feasible domain around the CMP. Without loss of generality, we will shift this problem so that the CMP corresponds to the origin in the discussion that follows.

A set of vectors $\{\tilde{\mathbf{x}}^1,\ldots,\tilde{\mathbf{x}}^k\}$ for $k \geq n+1$ is said to *positively span* $\mathbb{R}^n$ if any point in $\mathbb{R}^n$ may be reached via a linear combination of these vectors with non-negative coefficients. Since the $2n$ basis vectors $\{\mathbf{e}^1,\ldots,\mathbf{e}^n,-\mathbf{e}^1,\ldots,-\mathbf{e}^n\}$ positively span $\mathbb{R}^n$, a convenient test for whether or not the vectors $\{\tilde{\mathbf{x}}^1,\ldots,\tilde{\mathbf{x}}^k\}$ positively span $\mathbb{R}^n$ is to determine whether or not each vector in the set $\mathsf{E}=\{\mathbf{e}^1,\ldots,\mathbf{e}^n,-\mathbf{e}^1,\ldots,-\mathbf{e}^n\}$ can be reached by a positive linear combination of the vectors $\{\tilde{\mathbf{x}}^1,\ldots,\tilde{\mathbf{x}}^k\}$. That is, for each vector $\mathbf{e}\in\mathsf{E}$, a solution $\mathbf{z}$, with $z_i \geq 0$ for $i=1,\ldots,k$, to the equation $\tilde{X}\mathbf{z}=\mathbf{e}$ is sought, where $\tilde{X}=\begin{pmatrix}\tilde{\mathbf{x}}^1 & \ldots & \tilde{\mathbf{x}}^k\end{pmatrix}$. If such a $\mathbf{z}$ exists for each vector $\mathbf{e}\in\mathsf{E}$, then the vectors $\{\tilde{\mathbf{x}}^1,\ldots,\tilde{\mathbf{x}}^k\}$ positively span $\mathbb{R}^n$; if such a $\mathbf{z}$ does not exist, then the vectors $\{\tilde{\mathbf{x}}^1,\ldots,\tilde{\mathbf{x}}^k\}$ do not positively span $\mathbb{R}^n$.

Thus, testing a set of vectors to determine whether or not it positively spans $\mathbb{R}^n$ reduces simply to testing for the existence of a solution to $2n$ well-defined *linear programs* in standard form. Techniques to perform such tests, such as Matlab's `linprog`

algorithm, are well developed and readily available. Further, if a set of k vectors positiviely spans $\mathbb{R}^n$, it is a simple matter to check whether or not this set of vectors is also a positive basis of $\mathbb{R}^n$, if such a check is necessary, simply by checking whether or not any subset of $k-1$ vectors chosen from this set also positively span $\mathbb{R}^n$. Note that a positive basis with k vectors will necessarily have k in the range $n+1 \leq k \leq 2n$; the case with $k = n+1$ is referred to as a *minimal* positive basis, and the case with $k = 2n$ is referred to as a *maximal* positive basis.

3.2.2 Selecting a positive basis

In §6 of Chapter 2, we described how to enumerate all points which are nearest neighbors of the origin of a lattice (and thus, with the appropriate shift, all points which are nearest neighbors of any CMP on the lattice). In §3.2.1 above, we described how to test a subset of such points to see if the vectors from the origin to these points form a positive basis around the CMP. We now present a general algorithm to solve the problem of selecting a positive basis from the nearest-neighbors of the CMP using a minimal number of new poll points, while creating the maximum achievable angular uniformity between the vectors from the CMP to each of these points (that is, while minimizing the skewness of the resulting poll set). Note in Figure 3.2 that, as the number of nearest neighbors increases, the flexibility in solving this (apparently, NP-hard) problem also increases, though a perfectly distributed minimal positive basis (using $n+1$ points) is not always available. Ideally, for $m = 1$, the solution to the discrete Thomson problem will produce a positive basis with good angular uniformity; if it does not, we may successively increment m by one and try again until we succeed in producing a positive basis. We have studied three algorithms for solving this problem:

Algorithm A. If the kissing number τ of the lattice under consideration is relatively large (that is, if $\tau \gg n$; for example, for the Leech lattice Λ_{24}), then a straightforward algorithm can first be used to solve Thomson's problem on a continuous sphere in n dimensions. This can be done simply and quickly by fixing $q \geq 0$ repulsive particles at the prespecified lattice points, and initializing $n+m-q$ free repulsive particles on the sphere randomly. Then, at each iteration, a straightforward force-based algorithm may be used to move each free particle along the surface of the sphere a small amount

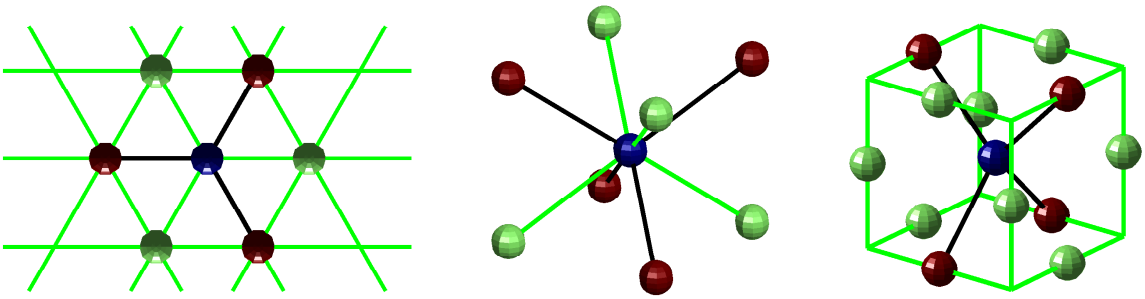

Figure 3.2: Various minimal positive bases (shown in red) around the origin (shown in blue) in the (left) triangular, (center) BCC, and (right) FCC lattices. Note that the triangular and BCC lattices each have two perfectly distributed minimal positive bases. In contrast, there are several choices for selecting a minimal positive basis in the FCC lattice, but none is perfectly distributed.

in the direction that the other particles are tending to push it, and iterating until the set of particles approaches an equilibrium. The free particle that is nearest to a nearest-neighbor lattice point around the CMP is then moved to said lattice point and fixed there, and the remaining free particles adjusted until they reach a new equilibrium. This adjust/fix/adjust/fix sequence is repeated until all particles are fixed at lattice points.

Algorithm B. If the kissing number τ of the lattice under consideration is relatively small (that is, if τ is *not* well over an order of magnitude larger than n), then it turns out to be more expedient to solve the discrete Thomson problem directly. To accomplish this, again taking the q presepecified repulsive particles as fixed, we initialize $n+m-q$ free repulsive particles randomly on $n+m-q$ nearest-neighbor lattice points around the CMP and then, at each iteration, move the two or three[4] free particles that are furthest from equilibrium in the force-based model described above (that is, those free particles which have the highest force component projected onto the surface of the sphere) into new positions selected from the available locations in such a way as to minimize the maximum force (projected onto the sphere) over the entire set of (fixed and free) particles. Though each iteration of this algorithm involves an exhaustive search for placing

[4]Moving more than two or three particles at a time in this algorithm makes each iteration computationally intensive, and has little impact on overall convergence of the algorithm, whereas moving only one at a time is found to significantly impede convergence to the optimal solution.

the two or three free particles in question, it converges quickly when τ is $O(100)$ or less.

Algorithm C. For intermediate kissing numbers τ, a hybrid approach may be used: a "good" initial distribution may be found using Algorithm *A*, then this distribution may be refined using Algorithm *B*.

In each of these algorithms, to minimize the number of new function evaluations required at each poll step, a check is first made to determine whether any previous function evaluations have already been performed on the nearest-neighbor lattice points around the CMP. If so, then particles are fixed at these locations, while the remaining particles are adjusted via one of the three algorithms described above. By so doing, previously-calculated function values may be used with maximum effectiveness during the polling procedure. When performing the poll step of a surrogate-based search, in order to orient the new poll set favorably (and, on average, exit the poll step quickly), a particle may also be fixed at the nearest neighbor point with the lowest value of the surrogate function; when polling, this poll point is thus evaluated first.

The iterative algorithms described above, though in practice quite effective, are not guaranteed to converge from arbitrary initial conditions to a positive basis for a given value of *m*, even if such a positive basis exists. To address this issue, if the algorithm used fails to produce a positive basis, the algorithm may be repeated using a new random starting distribution. Our numerical tests indicate that this repeated random initialization scheme usually generates a positive basis within a few initializations when such a positive basis indeed exists. Since at times, for a given *m*, there exists no configuration of the free particles on the nearest-neighbor lattice points that produces a positive basis, particularly when the previous function evaluations being leveraged are poorly configured, the number of new random initializations is limited to a prespecified value. Once this value is reached, *m* is increased by one and the process repeated. As the cost of each function evaluation increases, the user can increase the number of random initializations attempted using one of the above algorithms for each value of *m* in order to avoid the computation of extraneous poll points that might in fact be unnecessary if sufficient exploration by the discrete Thomson algorithm described above is performed.

Numerical tests have demonstrated the efficacy of this rather simple strategy, which reliably generates a positive basis while keeping computational costs to a min-

imum even when leveraging a relatively poor configuration of previous function evaluations and when working in relatively high dimension n. Additionally, the algorithm itself is independent of the lattice being used; the only inputs to the algorithm are the dimension of the problem, the locations of the nearest-neighbor lattice points, and the identification of those nearest-neighbor lattice points for which previous function evaluations are available.

3.2.3 Implementation of feasible domain boundaries

When implementing a global search in n dimensions, or even when implementing a local search on a function which is ill-defined for certain nonphysical values of the parameters (such as negative concentrations of chemicals), it is important to restrict the optimization algorithm to look only over a prespecified "feasible" region of parameter space. For simplicity, the present work assumes rectangular constraints on this feasible domain (that is, simple upper and lower bounds on each parameter value). An efficient n-dimensional lattice with packing radius ρ_n is used to quantize the interior of the feasible domain, efficient $(n-1)$-dimensional lattices with packing radius $\rho_{n-1} = \rho_n/2$ are used to quantize the portions of the boundary of the feasible domain with one active constraint (that is, the "faces"), efficient $(n-2)$-dimensional lattices with packing radius $\rho_{n-2} = \rho_n/4$ are used to quantize the portions of the boundary of the feasible domain with two active constraints (that is, the "edges"), etc. The present section describes how to search over the boundaries of the feasible domain, and how to move on and off of these boundaries as appropriate, while carefully restricting all function evaluations to the interior and boundary lattices in order to coordinate an efficient search.

We distinguish between two scenarios in which the polling algorithm as described thus far must be adjusted to avoid violating the $(n-1)$-dimensional boundaries[5] of the feasible domain. In the first scenario, the CMP is relatively far (that is, greater than ρ_n but less than $2\rho_n$) from the boundary of the feasible domain, and thus one or more of the poll points as determined by one of the algorithms proposed in §3.2.2 might land slightly outside this boundary. In this scenario, an effective remedy is simply to *eliminate* all lattice points which land outside of the feasible domain from the list of po-

[5] That is, the portions of the boundary with a single active constraint.

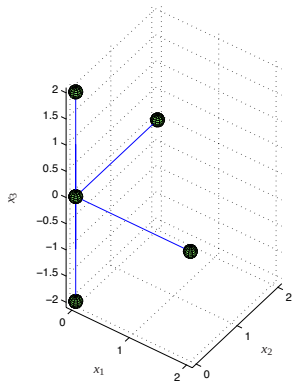

Figure 3.3: Constraint handling: a scenario in which a CMP at $\mathbf{x} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$ sits on an $(n-2) = 1$-dimensional edge of an $n = 3$-dimensional feasible region with bounds $x_1 \geq 0$ and $x_2 \geq 0$. Note that the feasible neighborhood of this edge is positively spanned by the nearest neighbors on the integer lattice, and that two additional vectors are added to the poll set to facilitate moving off of each of these active constraint boundaries.

tential poll points, and then to *augment* this restricted list of potential poll points with all lattice points on the nearby $(n-1)$-dimensional constraint surface which are less than $2\rho_n$ from the CMP. From this modified list of potential poll points, the poll set may be selected in the usual fashion using one of the algorithms described in §3.2.2.

In the second scenario, the CMP is relatively close (that is, less than ρ_n) to the boundary of the feasible domain. In this scenario, it is most effective simply to shift the CMP onto the nearest lattice point on the $(n-1)$-dimensional constraint surface. With the CMP on the feasible domain boundary, each poll step explores a minimum positive basis selected on the lattice quantizing the $(n-1)$-dimensional boundary and, in addition, polls an additional lattice point on the interior of the feasible domain to allow the algorithm to move back off this constraint boundary. Ideally, this additional point would be located on a inward-facing vector normal to the $(n-1)$-dimensional feasible domain boundary a distance ρ_n from the CMP; we thus choose the interior lattice point closest to this location.

Multiple active constraints are handled in an analogous manner (see Figure 3.3). In an n-dimensional optimization problem with $p \geq 2$ active constraints, the CMP is located on an active constraint "surface" of dimension $n-p$. An efficient $(n-p)$-

dimensional lattice with packing radius $\rho_{n-p} = \rho_n/2^p$ is used to quantize this active constraint surface, and a poll set is constructed by creating a positive basis selected from the points neighboring the CMP within the $(n-p)$-dimensional active constraint surface, together with p additional points located on the $(n-p+1)$-dimensional constraint surfaces neighboring the CMP. Ideally, these p additional points would be located on vectors normal to the $(n-p)$-dimensional active constraint surface a distance $\rho_{n-p+1} = \rho_n/2^{p-1}$ from the CMP; we thus choose the lattice points on the $(n-p+1)$-dimensional feasible domain boundaries closest to these locations.

In practice, it is found that, once an optimization routine moves onto $p \geq 1$ feasible domain boundaries, it only somewhat infrequently moves back off. To account for this, the p additional poll points mentioned in the previous paragraph are polled *after* the other poll points forming the positive basis within the $(n-p)$-dimensional active constraint surface.

3.2.4 Quantifying the skewness of positive bases

A final relevant metric of a lattice that relates to the performance of the corresponding lattice-based optimization is the deviation from perfect uniformity of the best minimal positive basis available on nearest-neighbor lattice points. The *best nearest-neighbor minimal positive basis skewness* of a lattice, s, is thus now defined as the ratio between the largest and the smallest angles between any two vectors in the best minimal positive basis available on nearest-neighbor lattice points, minus one. Therefore, $s = 0$ indicates a perfectly uniform minimal positive basis on nearest-neighbor lattice points, as exhibited by A_2 (see Figure 3.2a) and A_3^* (Figure 3.2b). In constrast, A_3 through A_8 all have $s = 0.3333$ (see, e.g., A_3 in Figure 3.2c).

Surprisingly, the best nearest-neighbor minimal positive basis skewness of E_8 is also $s = 0.3333$; one might initially expect it to be much smaller than this (indeed, one might hope that it would be fairly close to $s = 0$) due to the relatively large kissing number ($\tau = 240$) of this $n = 8$ lattice. Interestingly, the best nearest-neighbor positive basis of E_8 when using $n+2$ points (that is, instead of a minimal positive basis with $n+1$ points) is perfectly uniform. The tests reported in §3.5 thus use $n+2$ points instead of $n+1$ points when polling on the E_8 lattice.

A minimal positive basis on nearest-neighbor lattice points doesn't even exist on the $\mathbb{Z}^n$ lattice (indeed, a positive basis on nearest neighbors of the $\mathbb{Z}^n$ lattice requires a full $2n$ points). This was, in fact, a matter of significant inconvenience in previous work when using the Cartesian lattice as the default choice for such problems, as using a maximal positive basis rather than a minimal positive basis essentially doubles the cost of each complete poll step for large n. When developing a minimal positive basis for the $\mathbb{Z}^n$ lattice, it is thus common (see, e.g., Booker *et al.* 1999) to select the Cartesian unit vectors $\mathbf{e}^1$ through $\mathbf{e}^n$ and one additional "oddball" vector in the $(-1,-1,\ldots,-1)$ direction which is $\sqrt{n}$ longer. Note the "clustering" of the Cartesian unit vectors in directions generally opposite to the oddball vector. To quantify, the skewness of this minimal positive basis is $\cos^{-1}(-1/\sqrt{n})/(\pi/2)-1$, which in dimensions $n=2$ through 8 is given by 0.5, 0.3918, 0.3333, 0.2952, 0.2677, 0.2468, and 0.2301. Note that, while the skewness of the angular distribution of this minimal positive basis actually decreases gradually as the dimension of the problem increases, the ratio in lengths of the vectors to the nearest-neighbor lattice points and the oddball vector in this basis increases like $\sqrt{n}$ (that is, from 1.4142 in $n=2$ to 2.8284 in $n=8$). This is quite unfortunate, as it leads to a peculiar nonisotropic behavior of the optimization algorithm over parameter space (for further discussion on this point, see the sixth paragraph of §3.5.1). The tests reported in §3.5 use this peculiar minimum positive basis, with a long oddball vector, when polling on the $\mathbb{Z}^n$ lattice.

We now have all of the ingredients necessary to coordinate a GPS algorithm, as laid out in §3.1, with any of the lattices listed in Tables 2-3 of Chapter 2, while both reusing previous function evaluations and respecting sharp bounds on the feasible region of parameter space. Numerical testing of such an algorithm is reported in §3.5.

3.3 A review of the Kriging interpolation strategy

3.3.1 Interpolation - basic concepts

The purpose of the search step of an SMF algorithm is to interpolate, and extrapolate, the trends exhibited by the existing function evaluations in order to suggest new regions of parameter space, perhaps far from the CMP, where the function value

is anticipated, with some reasonable degree of probability, to be lower than that of the CMP. There are a variety of possible ways of accomplishing this; we leverage here the Kriging interpolation strategy (Krige 1951; Matheron 1963; Jones 2001; Rasmussen & Williams 2006).

The problem of interpolation is the problem of drawing a smooth curve through data points in order to estimate the function values in regions where the function itself has not yet been computed. The problem of interpolation, thus, necessarily builds on some hypothesis that models the function behavior in order to "connect the dots". The most common such model is a mechanical one, based on a thin piece of wood, or "spline", that is "bent" in order to touch all the data points; this mechanical model leads directly to the mathematical algorithm known as cubic spline interpolation. A perhaps equally valid hypothesis, which forms the foundation for the Kriging interpolation strategy, is to *model the underlying function as a realization, with maximum likelihood, of some stochastic process*. The stochastic model used in this approach is selected to be general enough to model a broad range of functions reasonably well, yet simple enough to be fairly inexpensive to tune appropriately based on the measured data. There are many such stochastic models which one can select; the simple stochastic model considered here leads to the easy-to-use interpolation strategy commonly referred to as ordinary Kriging.

3.3.2 Notation of statistical description

To begin, consider N points $\{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$, at which the function will ultimately be evaluated, and model the function's value at these N points with the random vector

$$\mathbf{f} = \begin{pmatrix} f(\mathbf{x}^1) \\ \vdots \\ f(\mathbf{x}^N) \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix}.$$

To proceed further, we need a clear statistical framework to describe this random vector.

The cumulative distribution function (CDF) of the random vector $\mathbf{f}$, denoted $d_{\mathbf{f}}(\underline{\mathbf{f}})$, is a mapping from $\underline{\mathbf{f}} \in \mathbb{R}^n$ to the real interval $[0,1]$ that monotonically increases in

each of the components of $\underline{\mathbf{f}}$, and is defined

$$d_{\mathbf{f}}(\underline{\mathbf{f}}) = P(f_1 \leq \underline{f}_1, f_2 \leq \underline{f}_2, \ldots, f_n \leq \underline{f}_n),$$

where $\underline{\mathbf{f}}$ is some particular value of the random vector $\mathbf{f}$ and $P(S)$ denotes a probability measure that the conditions stated in S are true. In the scalar case, for example, $d_f(1) = 0.6$ means that it is 60% likely that the random variable f satisfies the condition $f \leq 1$. For a random vector $\mathbf{f}$ whose CDF is modelled as being differentiable everywhere, the probability density function (PDF) $p_{\mathbf{f}}(\mathbf{f}') \geq 0$ is a scalar function of $\mathbf{f}'$ defined such that

$$d_{\mathbf{f}}(\underline{\mathbf{f}}) = \int_{-\infty}^{\underline{f}_1} \int_{-\infty}^{\underline{f}_2} \cdots \int_{-\infty}^{\underline{f}_n} p_{\mathbf{f}}(\mathbf{f}')\, d f_1' \, d f_2' \cdots d f_n' \quad \Leftrightarrow \quad p_{\mathbf{f}}(\mathbf{f}') = \frac{\partial^n d_{\mathbf{f}}(\underline{\mathbf{f}})}{\partial \underline{f}_1 \, \partial \underline{f}_2 \cdots \partial \underline{f}_n}\Big|_{\underline{\mathbf{f}}=\mathbf{f}'}.$$

For small $|\Delta \mathbf{f}'|$, the quantity $p_{\mathbf{f}}(\mathbf{f}')\Delta f_1' \Delta f_2' \cdots \Delta f_n'$ represents the probability that the random vector $\mathbf{f}$ takes some value within a small rectangular region centered at the value $\mathbf{f}'$ and of width $\Delta f_i'$ in each coordinate direction $\mathbf{e}_i$. Note that the integral of $p_{\mathbf{f}}(\mathbf{f}')$ over all possible values of $\mathbf{f}'$ is unity, that is

$$\int_{\mathbb{R}^n} p_{\mathbf{f}}(\mathbf{f}')\, d\mathbf{f}' = 1.$$

The expected value of a function $\mathbf{g}(\mathbf{f})$ of a random vector $\mathbf{f}$ is given by

$$\mathscr{E}\{\mathbf{g}(\mathbf{f})\} = \int_{\mathbb{R}^n} \mathbf{g}(\mathbf{f}')\, p_{\mathbf{f}}(\mathbf{f}')\, d\mathbf{x}'.$$

The expected value may be interpreted as the average of the quantity in question over many realizations. In particular, the mean $\overline{\mathbf{f}}$ and covariance $P_{\mathbf{f}}$ of the random vector $\mathbf{f}$ are defined as

$$\overline{\mathbf{f}} \triangleq \mathscr{E}\{\mathbf{f}\} = \int_{\mathbb{R}^n} \mathbf{f}'\, p_{\mathbf{f}}(\mathbf{f}')\, d\mathbf{f}', \qquad P_{\mathbf{f}} \triangleq \mathscr{E}\{(\mathbf{f}-\overline{\mathbf{f}})\,(\mathbf{f}-\overline{\mathbf{f}})^T\} = \int_{\mathbb{R}^n} (\mathbf{f}'-\overline{\mathbf{f}})\,(\mathbf{f}'-\overline{\mathbf{f}})^T\, p_{\mathbf{f}}(\mathbf{f}')\, d\mathbf{f}'.$$

3.3.3 Statistical modeling assumptions of ordinary Kriging

The PDF of the random vector $\mathbf{f} = \mathbf{f}_{n\times 1}$ in this analysis is modelled as Gaussian, and is thus restricted to the generic form

$$p_{\mathbf{f}}(\mathbf{f}') = \frac{1}{(2\pi)^{n/2}|P_{\mathbf{f}}|^{1/2}} \exp \frac{-(\mathbf{f}'-\overline{\mathbf{f}})^T P_{\mathbf{f}}^{-1} (\mathbf{f}'-\overline{\mathbf{f}})}{2}, \tag{3.1a}$$

where the covariance $P_{\mathbf{f}}$ is modelled as a constant σ^2, referred to as the variance, times a correlation matrix R whose $\{i,j\}$'th component r_{ij} is given by a model of the correlation of the random function f between points $\mathbf{x}^i$ and $\mathbf{x}^j$, where this correlation model $r(\cdot,\cdot)$ itself decays exponentially with the distance between points $\mathbf{x}^i$ and $\mathbf{x}^j$; that is,

$$P_{\mathbf{f}} \triangleq \sigma^2 R, \quad \text{where} \quad r_{ij} \triangleq r(\mathbf{x}^i,\mathbf{x}^j) \quad \text{and} \quad r(\mathbf{x},\mathbf{y}) \triangleq \prod_{\ell=1}^{n} \exp\Big(-\theta_\ell |x_\ell - y_\ell|^{p_\ell}\Big) \tag{3.1b}$$

for some yet-to-be-determined constants σ^2, $\theta_\ell > 0$, and $0 < p_\ell \leq 2$ for $\ell = 1,\ldots,n$. The mean $\bar{\mathbf{f}}$ in the Gaussian model (3.1a) is itself modelled as uniform over all of its components:

$$\bar{\mathbf{f}} \triangleq \mu \mathbf{1} \tag{3.1c}$$

for some yet-to-be-determined constant μ. There is extensive debate in the recent literature (see, e.g., Isaaks & Srivastava 1989; Rasmussen & Williams 2006) on the statistical modeling assumptions one should use in a Kriging model of this sort. It is straightforward to extend the present investigation to incorporate less restrictive Kriging models; the ordinary Kriging model is used here primarily due to its simplicity.

3.3.4 Optimization of the coefficients of the model

If the vector of observed function values is

$$\mathbf{f}^o = \begin{pmatrix} f_1^o \\ \vdots \\ f_N^o \end{pmatrix},$$

then the PDF corresponding to this observation in the statistical model proposed in (3.1) can be written as

$$p_{\mathbf{f}}(\mathbf{f}^o) = \frac{1}{(2\pi)^{n/2}(\sigma^2)^{n/2}|R|^{1/2}} \exp \frac{-(\mathbf{f}^o - \mu\mathbf{1})^T R^{-1} (\mathbf{f}^o - \mu\mathbf{1})}{2\sigma^2}. \tag{3.2}$$

The process of Kriging modeling boils down to selecting the parameters σ^2, θ_ℓ, p_ℓ, and μ in the statistical model proposed in (3.1) to maximize the PDF evaluated for the function values actually observed, $\mathbf{f} = \mathbf{f}^o$, as given in (3.2).

Maximizing $p_{\mathbf{f}}(\mathbf{f}^o)$ is equivalent to minimizing the negative of its log. Thus, for simplicity, consider

$$J = -\log[p_{\mathbf{f}}(\mathbf{f}^o)] = \frac{n}{2}\log(2\pi) + \frac{n}{2}\log(\sigma^2) + \frac{1}{2}\log(|R|) + \frac{(\mathbf{f}^o - \mu\mathbf{1})^T R^{-1} (\mathbf{f}^o - \mu\mathbf{1})}{2\sigma^2}. \tag{3.3}$$

Setting the derivatives of J with respect to μ and σ^2 equal to zero and solving, the optimal values of μ and σ^2 are determined immediately:

$$\mu = \frac{\mathbf{1}^T R^{-1} \mathbf{f}^o}{\mathbf{1}^T R^{-1} \mathbf{1}}, \qquad \sigma^2 = \frac{(\mathbf{f}^o - \mu \mathbf{1})^T R^{-1} (\mathbf{f}^o - \mu \mathbf{1})}{n}. \tag{3.4}$$

With these optimal values of μ and σ^2 applied, noting that the last term in (3.3) is now constant, what remains to be done is to minimize

$$J_1 = \frac{n}{2} \log(\sigma^2) + \frac{1}{2} \log(|R|) \tag{3.5}$$

with respect to the remaining free parameters[6] θ_ℓ and p_ℓ, where σ^2 is given as a function of R in (3.4) and R, in turn, is given as a function of the free parameters θ_ℓ and p_ℓ in (3.1b). This minimization must, in general, be performed numerically. However, the function J_1 is smooth in the parameters θ_ℓ and p_ℓ, so this optimization may be performed efficiently with a standard gradient-based algorithm, such as the nonquadratic conjugate gradient algorithm, where the gradient itself, for simplicity, may easily be determined via a simple finite difference or complex-step derivative approach.

Note that, after each new function evaluation, the Kriging parameters often adjust only slightly, and thus the previously-converged values of these parameters form a good initial guess for this gradient-based optimization algorithm. Note also that, while performing this optimization, the determinant of the correlation matrix occasionally reaches machine zero. To avoid the numerical difficulty that taking the log of zero would otherwise induce, a small $[O(10^{-6})]$ term may be added to the diagonal elements of R. By so doing, the Kriging predictor does not quite have the value of the sampled data at each sampled point; however, it remains quite close, and the algorithm is made numerically robust [Booker *et al*, 1999].

[6] To simplify this optimization, p_ℓ may be specified by the user instead of being determined via optimization; this is especially appropriate to do when the number of function evaluations N is relatively small, and thus there is not yet enough data to determine both the θ_ℓ and p_ℓ uniquely. If this approach is followed, $p_\ell = 1$ or 2 are natural choices; the case with $p_\ell = 1$ is referred to as an Ornstein-Uhlenbeck process, whereas the case with $p_\ell = 2$ is infinitely differentiable everywhere.

3.3.5 Using the tuned statistical model to predict the function value at new locations

Once the parameters of the stochastic model have been tuned as described above, the tuned Kriging model facilitates the computationally inexpensive prediction of the function value at any new location $\bar{\mathbf{x}}$. To perform this prediction, consider now the $N+1$ points $\{\mathbf{x}^1,\ldots,\mathbf{x}^N,\bar{\mathbf{x}}\}$, and model the function's value at these $N+1$ points with the vector

$$\bar{\mathbf{f}} = \begin{pmatrix} \mathbf{f} \\ f(\bar{\mathbf{x}}) \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \bar{f} \end{pmatrix},$$

where $\mathbf{f}$ is the $N \times 1$ random vector considered previously and $\bar{f}$ is the random scalar modeling the function at the new point. Analogous statistical assumptions as laid out in (3.1) are again applied, with the correlation matrix now written as

$$\bar{R} = \begin{bmatrix} R & \bar{\mathbf{r}} \\ \bar{\mathbf{r}}^T & 1 \end{bmatrix}, \qquad P_{\bar{\mathbf{f}}} \triangleq \sigma^2 \bar{R}, \tag{3.6}$$

where R is the $N \times N$ correlation matrix considered previously and, consistent with this definition, the vector $\bar{\mathbf{r}}$ is constructed with components

$$\bar{r}_i = r(\mathbf{x}^i, \bar{\mathbf{x}}), \quad \text{where} \quad r(\mathbf{x},\mathbf{y}) \triangleq \prod_{\ell=1}^{n} \exp\Big(-\theta_\ell |x_\ell - y_\ell|^{p_\ell} \Big).$$

Following Jones (2001), note by the matrix inversion lemma that $\bar{R}^{-1}$ may be written

$$\bar{R}^{-1} = \begin{bmatrix} R & \bar{\mathbf{r}} \\ \bar{\mathbf{r}}^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^{-1} + R^{-1}\bar{\mathbf{r}}(1-\bar{\mathbf{r}}^T R^{-1}\bar{\mathbf{r}})^{-1}\bar{\mathbf{r}}^T R^{-1} & -R^{-1}\bar{\mathbf{r}}(1-\bar{\mathbf{r}}^T R^{-1}\bar{\mathbf{r}})^{-1} \\ -(1-\bar{\mathbf{r}}^T R^{-1}\bar{\mathbf{r}})^{-1}\bar{\mathbf{r}}^T R^{-1} & (1-\bar{\mathbf{r}}^T R^{-1}\bar{\mathbf{r}})^{-1} \end{bmatrix}. \tag{3.7}$$

Keeping the parameter values σ^2, θ_ℓ, p_ℓ, and μ as tuned previously, we now examine the variation of the PDF in the remaining unknown random variable, $\bar{f}$. Substituting (3.6) and (3.7) into a PDF of the form (3.1a), we may write

$$p_{\bar{\mathbf{f}}}(\bar{\mathbf{f}}') = C_1 \cdot \exp \frac{-(\bar{\mathbf{f}}' - \mu\mathbf{1})^T \bar{R}^{-1} (\bar{\mathbf{f}}' - \mu\mathbf{1})}{2\sigma^2} = \ldots = C_2 \cdot \exp \frac{-[\bar{f}' - \hat{f}]^T [\bar{f}' - \hat{f}]}{2s^2}, \tag{3.8}$$

where, with a minor amount of algebraic rearrangement, the mean and variance of this

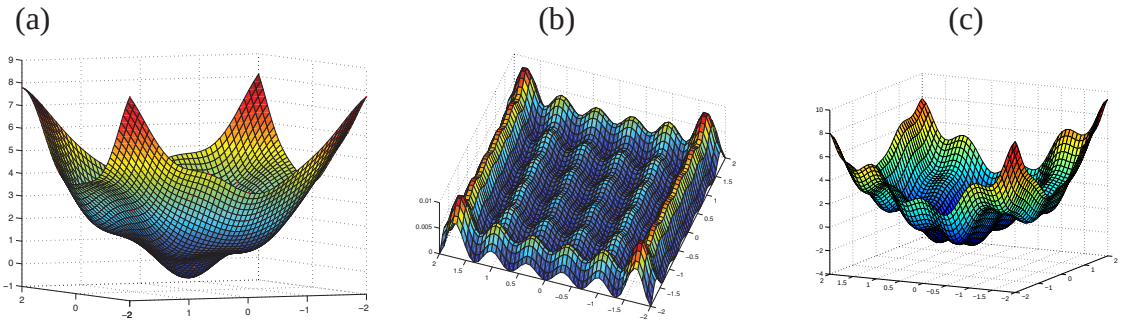

Figure 3.4: (a) The Kriging predictor, $\hat{f}(\mathbf{x})$, and (b) its associated uncertainty, $s^2(\mathbf{x})$, for a perturbed quadratic bowl sampled on a square grid of 7×7 points. (c) The corresponding MLI search function used for a global search in two dimensions (see §3.4).

scalar Gaussian distribution modeling the random scalar $\bar{f}$ work out to be[7]

$$\hat{f}(\bar{\mathbf{x}}) = \mathscr{E}\{f(\bar{\mathbf{x}})\} = \mathscr{E}\{\bar{f}\} = \mu + \mathbf{r}^T R^{-1}(\mathbf{f}^o - \mu\mathbf{1}), \tag{3.9a}$$

$$s^2(\bar{\mathbf{x}}) = \mathscr{E}\{[f(\bar{\mathbf{x}}) - \hat{f}]^2\} = \mathscr{E}\{[\bar{f} - \hat{f}]^2\} = \sigma^2(1 - \mathbf{r}^T R^{-1}\mathbf{r}). \tag{3.9b}$$

Equations (3.9a)-(3.9b) give the final formulae for the Kriging predictor, $\hat{f}(\bar{\mathbf{x}})$, and its associated uncertainty, $s^2(\bar{\mathbf{x}})$.

When applied numerically to a representative test problem, as expected, the Kriging predictor function, which we denote $\hat{f}(\bar{\mathbf{x}})$, interpolates [that is, it goes through every observed function value at points $\bar{\mathbf{x}} = \mathbf{x}^1$ to $\bar{\mathbf{x}} = \mathbf{x}^N$], whereas the uncertainty function, denoted $s^2(\bar{\mathbf{x}})$, is zero at each sampled point, and resembles a Gaussian "bump" between these sampled points, as seen in Figure 3.4. Note that, once the parameters of the statistical model have been determined, as described in §3.3.4, the formula (3.9a)-(3.9b) for the Kriging predictor $\hat{f}(\bar{\mathbf{x}})$ and its corresponding uncertainty $s^2(\bar{\mathbf{x}})$ at any test point $\bar{\mathbf{x}}$ is computationally quite inexpensive[8].

[7] An alternative interpretation of this process models the constant μ itself as a stochastic variable rather than as a constant. Following this line of reasoning ultimately gives the same formula for the predictor $\hat{f}(\bar{\mathbf{x}})$ as given in (3.9a), and a slightly modified formula for its associated uncertainty,

$$s^2(\bar{\mathbf{x}}) = \sigma^2\left(1 - \mathbf{r}^T R^{-1}\mathbf{r} + \frac{(1 - \mathbf{r}^T R^{-1}\mathbf{r})^2}{\mathbf{1}^T R^{-1}\mathbf{1}}\right). \tag{3.9b'}$$

Which formula [(3.9b) or (3.9b')] is used in the present model is ultimately a matter of little consequence as far as the overall derivative-free optimization algorithm is concerned; we thus prefer the form given in (3.9b) due to its computational simplicity.

[8] Note that, for maximum efficiency, R^{-1} should be saved between function evaluations and reused for every new computation of $\hat{f}$ and s^2 required.

3.4 A review of global optimization strategies leveraging Kriging-based interpolation

The previous section reviewed the Kriging interpolation strategy which, based on a sparse set of observed function values $f^o(\mathbf{x}^i)$ for $i = 1,\ldots,N$, develops a function predictor $\hat{f}(\mathbf{x})$ and a model of the uncertainty $s^2(\mathbf{x})$ associated with this prediction for any given set of parameter values $\mathbf{x}$. Leveraging this Kriging model, an efficient search algorithm can now be developed for the derivative-free optimization algorithm summarized in §3.1.

The effectiveness of the various Kriging-based search strategies which one might propose may be tested by applying them repeatedly to simple test problems via the following procedure:

- a search function $J(\mathbf{x})$ is first developed based on a Kriging model fit to the existing function evaluations,
- a gradient-based search is used to minimize this (computationally inexpensive, smoothly-varying) search function,
- the function $f(\mathbf{x})$ is sampled at the point $\tilde{\mathbf{x}}$ which minimizes the search function[9],
- the Kriging model is updated, and the search is repeated.

In the present work, we consider a scalar test problem with multiple minima, $f(x) = \sin(x) + x^2$, on the interval $x \in [-10, 10]$, and use four starting points to initialize the search $x = -10$, $x = -5.2$, $x = 6$, and $x = 10$. Ineffective search strategies will not converge to the global minimum of $f(x)$ in this test, and may not even converge to a local minimum. More effective search strategies converge to the global minimum following this approach, and the number of function evaluations required for convergence indicates the effectiveness of the search strategy used.

Perhaps the most "obvious" strategy to use in such problems is simply fitting a Kriging model to the known data, then searching the Kriging predictor itself, $J(\mathbf{x}) = \hat{f}(\mathbf{x})$, for its minimum value. This simple approach has been implemented in a variety of examples with reasonably good results (see Booker *et al*, 1999). However, as shown

[9]For the moment, to focus our attention on the behavior of the search algorithm itself, no underlying grid is used to coordinate the search in order to keep function evaluations far apart.

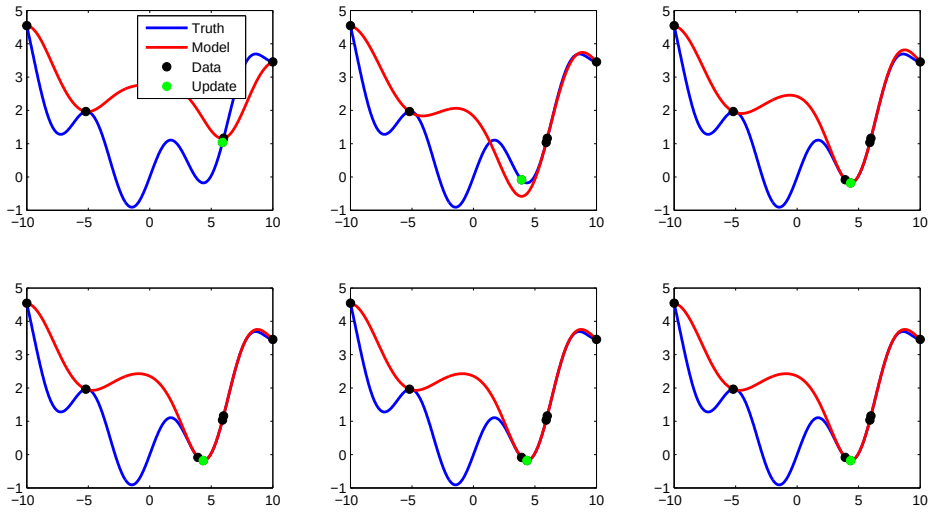

Figure 3.5: Convergence of a search algorithm minimizing the Kriging predictor, $J(\mathbf{x}) = \hat{f}(\mathbf{x})$, at each iteration. This algorithm does not necessarily converge to even a local minimum, and in this example has stalled, far from the global minimum, after six iterations.

clearly in Figure 3.5, this approach can easily break down. The Kriging predictor does not necessarily model the function accurately, and its minimization fails to guarantee convergence to even a local minimum of the function $f(\mathbf{x})$. This observed fact can be motivated informally by identifying the Kriging predictor as an *interpolating* function which only under extraördinary conditions predicts a function value significantly lower than all of the previously-computed function values; under ordinary conditions, a strategy of minimizing the predictor will thus often stall in the vicinity of the previously-evaluated points.

To avoid the shortcomings of a search defined solely by the minimization of the predictor, another strategy explored by Booker *et al* (1999) is to evaluate the function at *two* points in parameter space during the search: one point chosen to minimize the predictor, and the other point chosen to maximize the predictor uncertainty. Such a heuristic provides a guarantee of global convergence, as the seach becomes dense in the parameter space as the total number of function evaluations, N, approaches infinity (see §3.1). However, this approach generally does not converge quickly as compared

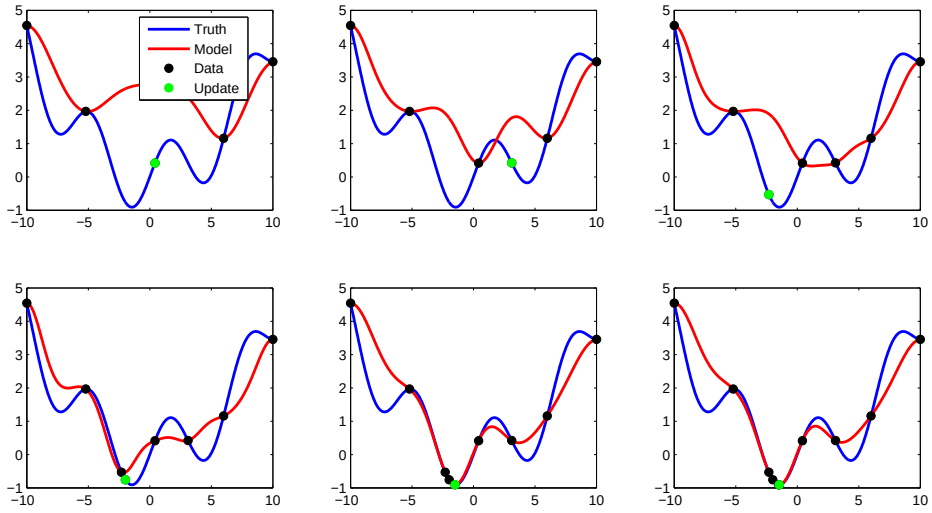

Figure 3.6: Convergence of a search algorithm minimizing $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ at each iteration, taking $c = 1$. Note that the global minimum is found after just a few iterations. However, global convergence is not guaranteed.

with the improved methods described below, as the extra search point has no component associated with the predictor, and is thus often evaluated in relatively "poor" regions of parameter space.

We are thus motivated to develop a more flexible strategy to explore *slightly away* from the minima of the predictor. To achieve this, consider the minimization of $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$, where c is some constant (see Cox & John 1997 and Jones 2001). A search coordinated by this function will tend to explore regions of parameter space where both the predictor of the function value is relatively low *and* the uncertainty of this prediction in the Kriging model is relatively high. With this strategy, the search is driven to regions of higher uncertainty, with the $-c \cdot s^2(\mathbf{x})$ term in $J(\mathbf{x})$ tending to cause the algorithm to explore away from previously evaluated points. Additionally, minimizing $\hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ allows the algorithm to explore the vicinity of *multiple* local minima in successive iterations in order to determine, with an increasing degree of certainty, which local "bowl" in fact has the deepest minimum. The parameter c provides a natural means to "tune" the degree to which the search is driven to regions of higher uncertainty, with smaller values of c focusing the search more on refining the vicinity

of the lowest function value(s) already found, and larger values of c focusing the search more on exploring regions of parameter space which are still relatively poorly sampled. This parameter may tuned based on knowledge of the function being minimized: if the function is suspected to have multiple minima, c can be made relatively large to ensure a more exploratory search, whereas if the function is suspected of having a single minimum, c can be made relatively small to ensure a more focused search in the vicinity of the CMP. For an appropriate intermediate value of c, the resulting algorithm is often quite effective at both global exploration and local refinement of the minimum, as illustrated in Figure 3.6. The strategy of searching $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ also extends naturally to multiple dimensions, as illustrated for a two-dimensional problem in Figure 3.4c. Note also that, in the spirit of Booker *et al* (1999) [who effectively suggested, in the present notation, exploring based on both $c = 0$ and $c \to \infty$ at each search step], one can perform a search using multiple but finite values of c at each search step, returning a set of points designed to focus, to varying degrees, on the competing objectives of global exploration and local refinement. If at each search step k at least one point is included which minimizes $\hat{f}(\mathbf{x}) - c_k \cdot s^2(\mathbf{x})$ for a value of c_k which itself approaches ∞ as $k \to \infty$, then the search drives at least some new function evaluations sufficiently far from the existing points that the function evaluations eventually become dense over the feasible domain, thus guaranteeing global convergence. Thus, an $\hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ search, when used properly, can indeed be used in a globally convergent manner.

Minimizing $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ is not the only strategy to take advantage of the estimate of the uncertainty of the predictor provided by the Kriging model. Another effective search strategy involves maximizing the probability of achieving a target level of improvement below the current CMP; this is called the *maximum likelihood of improvement* (*MLI*) approach [see Kushner 1964, Stuckman 1988, Perttunen 1991, Elder 1992, and Mockus 1994]. If the current CMP has a function value $f_{\min}$, then this search strategy seeks that $\mathbf{x}$ for which the probability of finding a function value $f(\mathbf{x})$ less than some prespecified target value f_{target} [that is, for which $f(\mathbf{x}) \leq f_{\text{target}} < f_{\min}$] is maximized in the Kriging model. If $f(\mathbf{x})$ is known to be a positive function, a typical target value in this approach is $f_{\text{target}} = (1 - \delta) f_{\min}$, where δ may be selected somewhere in the range of 0.01 to 0.2. As for the parameter c discussed in the previous paragraph,

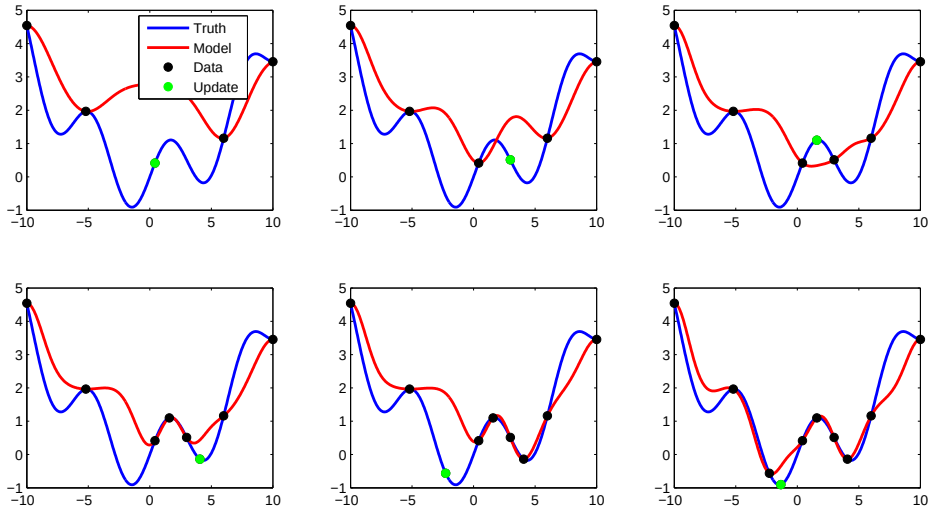

Figure 3.7: MLI search with a target $T = 10\%$. Note convergence to global minimum, as well as exploratory nature of the search which guarantees global convergence.

the parameter δ in this strategy tunes the degree to which the search is driven to regions of higher uncertainty, with smaller values of δ focusing the search more on refining the vicinity of the lowest function value(s) already found, and larger values of δ focusing the search more on exploring regions of parameter space which are still relatively poorly sampled. As seen in Figure 3.7, the MLI search offers performance similar to the $\hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ method discussed previously. In contrast with the $\hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ approach, even for a fixed (finite) value of δ, the MLI approach eventually drives the function evaluations far enough away from existing points that the function evaluations eventually become dense over the feasible domain, thus guaranteeing global convergence. Thus, the MLI approach is inherently globally convergent.

Even more sophisticated search strategies can also be proposed, as reviewed elegantly by Jones (2001). However, the simplicity, flexibility, and performance given by the strategy of maximizing the MLI renders this approach as adequate for our testing purposes here.

Since both the $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ search function and the MLI search function are inexpensive to compute, continuous, and smooth, but in general have multiple

minima, an efficient gradient-based search, initialized from several well-selected points in parameter space, may be used to to minimize them. As the uncertainty $s^2(\mathbf{x})$ goes to zero at each sample point, $J(\mathbf{x})$ will tend to dip between each sample point. Thus, a search is initialized on $2n \cdot N$ total points forming a positive basis near (say, at a distance of $\rho_n/2$) to each of the N sample points, and each of these starting points is marched to a local minima of the search function using an efficient gradient-based search (which is constrained to remain within the feasible domain of $\mathbf{x}$). The lowest point of the paths so generated will very likely be the global minima of the search function. For simplicity, the necessary gradients for this search may be computed via a simple second-order central finite difference scheme applied to the Kriging model, though more sophisticated and efficient approaches are also possible.

3.5 Results

Putting everything together, we now develop and test what we identify as the *Lattice Based Derivative-free Optimization via Global Surrogates* (*LABDOGS*) algorithm. This algorithm consists of an SMF-based optimization (see §3.1) coordinated by uniform n-dimensional lattices (see §3.2, and further discussion in Chapter 2 of this study) while leveraging a Kriging interpolant (see §3.3) to perform an efficient global search based on the MLI search function(see §3.4). The full algorithm has been implemented in an efficient numerical code and is tested in this section in $n = 2$ to $n = 8$ dimensions using the $\mathbb{Z}^n$, A_n, and E_8 lattices to coordinate the search, and is applied here to: Shifted quadratic bowls:

$$f_Q(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^o)^T A(\mathbf{x} - \mathbf{x}^o)$$

Shifted Rosenbrock functions:

$$f_R(\mathbf{x}) = \sum_{i=0}^{n-1} \{[1 - (x_i - x_i^o)]^2 + (-1)^n 500[(x_{i+1} - x_{i+1}^o) - (x_i - x_i^o)^2]^2$$

The Branin function:

$$f_B(\mathbf{x}) = [1 - 2x_2 + 0.05\sin(4\pi x_2) - x_1]^2 + [x_2 - 0.5\sin(2\pi x_1)]^2$$

The "T_1" function:

$$f_M(\mathbf{x}) = \sin(5x_1) + \sin(5x_2) + 0.02[(5x_1 + 1.5)^2 + (5x_2 + 1.5)^2]$$

Note that the first two test functions are *n*-dimensional and have unique minima, whereas the last two test functions are 2-dimensional and have multiple minima.

3.5.1 SP applied to convex functions

To test the hypothesis that the efficiency of a pattern search is significantly affected by the packing efficiency and/or the nearest-neighbor distribution of the lattices which coordinate it, a large number of SP optimizations were first performed on randomly-shifted quadratic bowls to gather and compare statistical data on the performance of $\mathbb{Z}^n$-based, A_n-based, and E_8-based SP optimizations. The positive-definite matrices $A > 0$ and offsets $\mathbf{x}^o$ defining the quadratic bowls to be minimized, as well as the starting points used in the searches, were selected at random for every set of tests, and the initial $\mathbb{Z}^n$, A_n, and E_8 lattices were scaled such that the initial number of points per unit volume of parameter space was identical.

The $\mathbb{Z}^n$-based, A_n-based, and E_8-based SP algorithms were run from the same starting points on the same quadratic test functions to the same level of convergence. Note that several of the significant built-in acceleration features of the full LABDOGS code were in fact turned off for this baseline comparison. Most notably, complete polls were performed (that is, the poll steps were not terminated immediately upon finding a lower CMP), and no attempt was made to reuse previously-computed points when forming each successive poll set, or to orient optimally any given poll set. In fact, the angular distribution of the poll set around the CMP was fixed from one step to the next in these initial tests.

Two quantitative measures of the relative efficiency of the optimization algorithms to be tested are now defined. The metric p is defined as the *percentage of runs* in which the lattice-based algorithm requires fewer function evaluations than does the $\mathbb{Z}^n$-based algorithm to converge 99.99% of the way from the initial value of $J(\mathbf{x})$ to the optimal value of $J(\mathbf{x})$ [which, in these test problems, is easy to compute analytically]. The metric r is defined as the *ratio of the average number of function evaluations* re-

Table 3.3: Performance comparison between the A_n-based SP algorithm and the $\mathbb{Z}^n$-based SP algorithm applied to randomly shifted quadratic bowls for $n = 2$ to 8. It is seen that the A_8-based SP algorithm outperformed the $\mathbb{Z}^8$-based SP algorithm 85% of the time, and on average required 30% as many function evaluations to reach the same level of convergence.

n	2	3	4	5	6	7	8
p	74.77	81.32	84.03	84.53	84.43	84.56	85.28
r	0.4290	0.4161	0.3273	0.3585	0.3150	0.3345	0.3060

Table 3.4: Performance comparison between the E_8-based SP algorithm and the $\mathbb{Z}^8$-based SP algorithm applied to randomly shifted quadratic bowls. It is seen that the E_8-based SP algorithm outperformed the $\mathbb{Z}^8$-based SP algorithm 91% of the time, and on average required 15.5% as many function evaluations to reach the same level of convergence, thus offering nearly twice the performance of A_n.

n	8
p	90.65
r	0.1554

quired for the lattice-based algorithm to converge 99.99% of the way from the initial value of $J(\mathbf{x})$ to the optimal value of $J(\mathbf{x})$ divided by the average number of function evaluations needed for the $\mathbb{Z}^n$-based algorithm to converge the same amount.

The p and r measures described above (averaged over 5000 runs for each value of n) were calculated in the case of the A_n lattice (for $n = 2$ to $n = 8$) and the E_8 lattice, and are reported in Tables 3.3 and 3.4. Note that values of p over 50% and values of r less than 1 indicate that, on average, the lattice-based SP algorithm outperforms the $\mathbb{Z}^n$-based SP algorithm, with p quantifying how often and r quantifying how much.

Note in Table 2.1 that the "best" lattice in $n = 2$ and $n = 3$, accoring to several standard metrics, is A_n; however, as the dimension of the problem increases, several other lattices become available, and that by $n = 8$ the E_8 lattice appears to be the best choice. This observation is consistent with the numerical results reported in Tables 3.3 and 3.4, which indicates that the A_n-based optimizations provided a consistent and

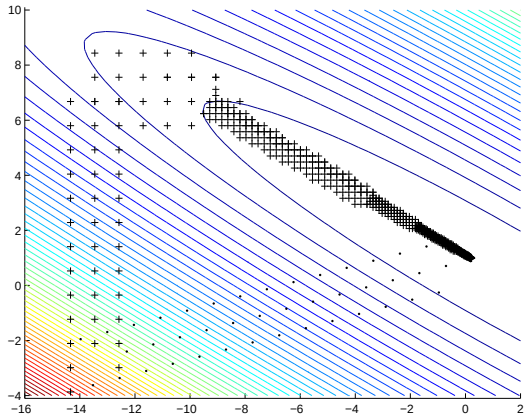

Figure 3.8: Typical paths taken by the A_2-based SP algorithm (dots) and the $\mathbb{Z}^2$-based SP algorithm (+) on a randomly-shifted quadratic bowl.

substantial improvement over the $\mathbb{Z}^n$-based optimizations over the entire range $n = 2$ to 8, and that, in $n = 8$, the E_8-based optimization significantly outperformed the A_8-based optimization.

The mechanism by which the lattice-based SP algorithms outperform the $\mathbb{Z}^n$-based SP algorithm on quadratic test problems is now examined in detail. As described previously, the $\mathbb{Z}^n$ minimal positive basis vectors are distributed with poor angular uniformity and can not be selected on nearest-neighbor lattice points. When the optimal descent direction is poorly approximated by these $n+1$ vectors (such as when the optimal descent direction is configured somewhere approximately midway between the oddball vector and one of the Cartesian unit vectors), the search path must "zig-zag" to move towards the actual minimum. If the local curvature of the function is small compared to the current lattice spacing, then the search algorithm must take several steps in a rather poor direction before it must eventually turn back down the "valley floor", as illustrated by the path of the $\mathbb{Z}^n$-based SP algorithm in Figure 3.8. Once in this valley, the lattice spacing must be diminished such that each step of the "zig-zag" path required to proceed down the valley floor in fact decreases the function; this leads to otherwise unnecessary lattice refinement and thus very slow progress by the SP algorithm. This effect is exacerbated when the vectors of the poll set are of substantially different length, as the entire set of vectors must be scaled down until movement along the direction of

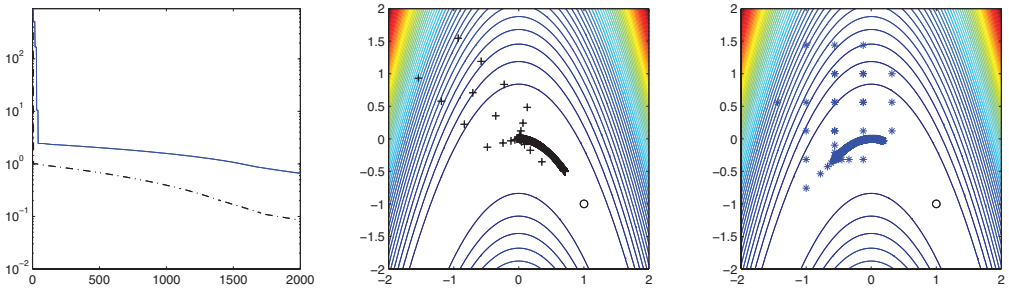

Figure 3.9: A sample SP minimization comparing the A_n-based case (dash-dot line at left and black $+$ at center) with the $\mathbb{Z}^n$-based case (solid line at left and blue $*$ at right) on a randomly shifted Rosenbrock function. Note the superior convergence rate of the A_n-based approach (as illustrated in the convergence plot at left), resulting in further progress toward the minimum at $[1, -1]$ (as illustrated in the subfigures at center and right).

the longest poll vector during this zig-zagging motion still decreases the function. This leads to the poor convergence behavior demonstrated by the $\mathbb{Z}^n$-based SP algorithm along the narrow valley floor of the quadratic bowl indicated in Figure 3.8. Of course, the present arguments are statistical in nature, and in specific cases either the A_n-based SP algorithm or the $\mathbb{Z}^n$-based SP algorithm will sometimes get "lucky" and converge remarkably quickly. However, it is clear that the optimal descent direction at any given iteration is more likely to be "far" from the poll vectors when the poll set is distributed with poor angular uniformity.

The A_n-based and $\mathbb{Z}^n$-based SP algorithms were also applied to a randomly-shifted Rosenbrock function in a similar fashion. Figure 3.9 demonstrates a typical case, indicating the respective rates of convergence of the two SP algorithms. The A_n-based SP algorithm demonstrates a substantially improved convergence rate compared to the $\mathbb{Z}^n$-based SP algorithm.

These results demonstrate that the efficiency of the SP portion of a pattern search can be substantially improved simply by implementing a more efficient lattice to discretize parameter space.

3.5.2 LABDOGS applied to Rosenbrock functions

To test the hypothesis that the efficiency of the full LABDOGS algorithm is significantly affected by the choice of the lattices which coordinate it, a more demanding test than a quadratic bowl is required. We thus consider here the application of the full LABDOGS algorithm to randomly shifted Rosenbrock functions. The "valley" in which the minimum of the Rosenbrock function lies is narrow, curved, and relatively flat (that is, with a vanishing second derivative) along the bottom. This makes it a difficult test case for any SMF-like algorithm to approximate with a surrogate function of sufficient accuracy to be particularly useful along the valley floor, other than simply to indicate where the function evaluations are currently relatively sparse. In other words, both the search and poll components of the LABDOGS algorithm are put to the test when searching along the valley floor of the Rosenbrock function.

Two comparisons of the efficiencies of the A_n-based and $\mathbb{Z}^n$-based LABDOGS algorithms (using $c = 5$) applied to randomly shifted Rosenbrock functions are reported here. As in the SP tests described previously, the initial A_n and $\mathbb{Z}^n$ lattices were scaled appropriately so as to be of the same initial density.

Recall in the SP tests the metric p, which quantified *how often* the lattice-based method outperformed the Cartesian-based method, and the metric r, which quantifying *how much* the lattice-based method outperformed the Cartesian-based method. In this section, we use two similar metrics, $\bar{p}$ and $\bar{r}$, but now terminate each optimization after a particular number of iterations rather than after convergence to a given percentage of the (known) optimal solution. Specifically, the metric $\bar{p}$ is defined as the percentage of runs in which the A_n-based LABDOGS algorithm converged further than did the $\mathbb{Z}^n$-based LABDOGS algorithm after 300 function evaluations, whereas the metric $\bar{r}$ is defined as the ratio of the average function value to which the A_n-based LABDOGS algorithm converged after 300 function evaluations divided by the average function value to which the $\mathbb{Z}^n$-based LABDOGS algorithm converged after 300 function evaluations. The results for $n = 2$ to 5 (averaged over 200 runs for $n = 2$, 3, and 4, and 100 runs for $n = 5$) are reported in Table 3.5. Note that values of $\bar{p}$ over 50% and values of $\bar{r}$ less than 1 indicate that, on average, the lattice-based LABDOGS algorithm outperforms the $\mathbb{Z}^n$-based LABDOGS algorithm, with $\bar{p}$ quantifying how often and $\bar{r}$ quantifying how much. It is

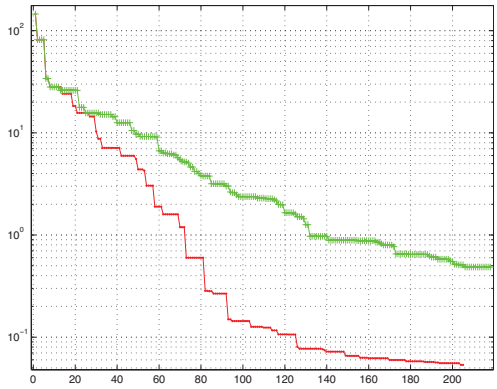

Figure 3.10: Convergence of the LABDOGS code using A_n (red) vs $\mathbb{Z}^n$ (green), on an $n = 6$ Rosenbrock function.

Table 3.5: Performance comparison between the A_n-based LABDOGS algorithm and the $\mathbb{Z}^n$-based LABDOGS algorithm applied to randomly shifted Rosenbrock functions. For $n = 2$, it is seen that the A_n-based SP algorithm outperformed the $\mathbb{Z}^n$-based SP algorithm about 64% of the time, and on average converged to a function value 65% better using the same number of function evaluations.

n	2	3	4	5
$\bar{p}$	64.0	56.0	63.0	68.0
$\bar{r}$	0.651	0.699	0.773	0.758

seen that the A_n-based LABDOGS algorithm consistently and significantly outperforms the $\mathbb{Z}^n$-based LABDOGS algorithm.

Figure 3.10 compares the convergence of the A_n-based and $\mathbb{Z}^n$-based LABDOGS algorithms on a representative realization of the Rosenbrock function in $n = 6$. The convergence of the two algorithms are similar in behavior during the first 20 iterations, during which they share a nearly identical search, with the differences between the two becoming more and more apparent as convergence is approached. Initially, the poll steps return much smaller improvements than the search steps. Once the surrogate model adequately represents the walls of the Rosenbrock function, thereby identifying the "valley floor", the search becomes less effective, and both algorithms rely more heavily on the polling algorithm to identify the minimum.

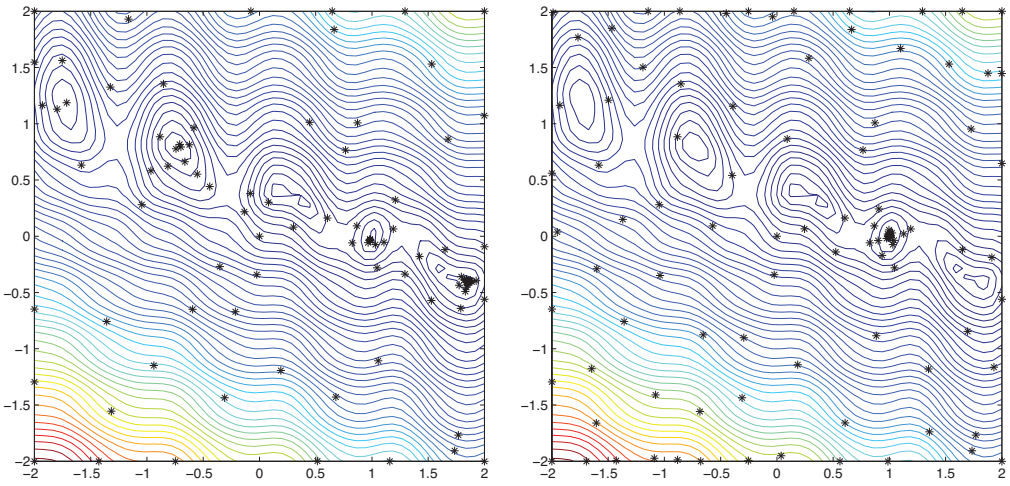

Figure 3.11: Points evaluated by the LABDOGS algorithm when exploring the Branin function (with multiple minima), with (left) $c = 2$ and (right) $c = 10000$. Note the more "focused" sampling when c is small and the more "exploratory" sampling when c is large.

3.5.3 LABDOGS applied to Branin and T_1 —demonstrating global exploration with local refinement

Thus far, only functions with unique minima have been explored. As the LABDOGS algorithm has the capability to locate and explore multiple local minima in an attempt to identify and refine an estimate of the global minimum, some searches were performed on two test functions with multiple minima, Branin and T_1, to demonstrate this capability.

On the interval $-2 < x < 2, -2 < y < 2$, the Branin function has five local minima. As seen in Figure 3.11, with the search parameter $c = 2$, the LABDOGS algorithm does an excellent job of locating and exploring all of these local minima, eventually converging to an accurate estimate of the global minimum. With $c = 10000$, the search tends to be more "space-filling", acting at each step to reduce the maximum uncertainty of the Kriging surrogate. It is clearly evident that, as the number of function evaluations gets large in the $c = 10000$ case, this search will tend to explore nearly uniformly over the entire feasible domain. [In the limit that c is infinite, the function evaluations become dense as $N \to \infty$, thereby assuring global convergence.] However, for a small

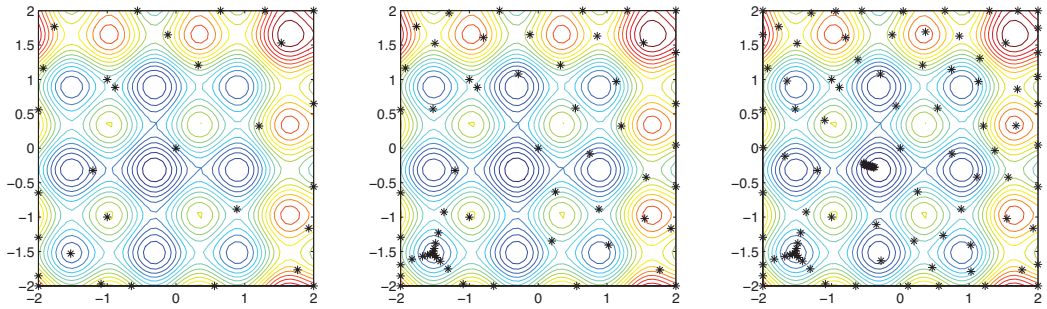

Figure 3.12: Points evaluated by the LABDOGS algorithm when exploring the T_1 function (with multiple minima) with $c = 1000$ after (left) 30 function evaluations, (center) 60 function evaluations, and (right) 100 function evaluations. Note (after 30 function evaluations) that the LABDOGS algorithm initially identifies and converges to a local minimum near the lower-left corner. Ultimately (after 100 function evaluations), the LABDOGS algorithm successfully identifies a refined estimate of the global minimum.

number of total function evaluations N [which should be the primary problem of interest if function evaluations are expensive!], the strategy with smaller c in fact identifies and refines the estimate of the global minimum point much sooner, as the case with large c wastes a lot of computational effort reducing the uncertainty of the surrogate in areas predicted to have poor function values.

Similar behavior can be seen for the T_1 test function in Figure 3.12. Initially, the algorithm happens upon the local minimum in the lower-left corner of the feasible domain. With its exploratory function evaluations, however, the algorithm ultimately identifies and refines its estimate of the global minimum.

While these results indicate encouraging global exploration, further testing of the LABDOGS algorithm on nonconvex functions is certainly warranted, particularly in high-dimensional problems. In particular, further refinement of the algorithm to provide the most robust combination of "focused" and "exploratory" sampling remains to be performed; however, the present results clearly demonstrate the capability and flexibility of the LABDOGS algorithm to strike this balance while maintaining maximum computational efficiency.

3.6 Conclusions

The present work proposes a new algorithm, dubbed LABDOGS, for derivative-free optimization formed via the tight integration of

- the efficient SMF algorithm (see §3.1) for a surrogate-based search coordinated by an underlying grid, in order to keep function evaluations far apart until convergence is approached,
- a uniform "grid" selected from those available in lattice theory (see §3.2 and further discussion in Chapter 2 of this study) to coordinate such an optimization algorithm, in order to reduce the average quantization error of a grid of a given density and to better distribute the poll points during the poll step, and
- a highly effective search algorithm, leveraging a Kriging interpolant (see §3.3) to construct the MLI search function combining both the function predictor and a model of its associated uncertainty, in order to provide a flexible combination of global exploration and local refinement during the search (see §3.4).

The numerical results achieved via this algorithm (see §3.5) indicate effective convergence of the resulting algorithm on a range of benchmark optimization problems, and reveal a clear advantage for using an efficient lattice derived from an n-dimensional sphere packing to coordinate such a search, rather than the heretofore default choice, $\mathbb{Z}^n$, which is simply untenable in light of the clear advantages of using alternative lattices which are, quantifiably, both more uniform and have a more favorable distribution of nearest neighbors, especially as the dimension of the optimization problem is increased.

The flexible numerical code we have developed which implements the LABDOGS algorithm has been written from scratch, and each subroutine of the code has been scrutinized to maximize its overall efficiency for systems with expensive function evaluations.

Much interesting work remains to be done. The possible applications of such a derivative-free optimization code are quite broad. Natural extensions of the algorithm proposed herein include the implementation and testing of a variety of lattices, more sophisticated versions of Kriging interpolation, and appropriate penalities for online parameter tuning; such extensions are all well underway, and will be reported in future

work.

3.7 Acknowledgements

Chapter 3, in full, is a reprint of the material as it has been submitted to the Journal of Global Optimization. Belitz, P., Bewley, T., 2011. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Lattice-based Mesh Adaptive Direct Search (Λ-MADS)

4.1 Introduction

This chapter introduces and tests Λ-MADS, a new variant of the Mesh Adaptive Direct Search (MADS) class of derivative-free optimization algorithms for constrained nonsmooth functions that is built on maximally uniform *lattices* Λ_n, rather than Cartesian grids $\mathbb{Z}^n$, as the underlying mesh used to coordinate the exploration of parameter space. When a poll step fails to find a mesh point with a better function value than that of the current candidate minimum point (CMP), in addition to reorienting the poll set, a mesh refinement of a factor of 2 (rather than a factor of 4, as used in previous MADS implementations) is performed in Λ-MADS; slowing the refinement of the mesh in this manner as the iteration proceeds is found to increase the rate of convergence, as an appropriately-coarse underlying mesh is valuable in generalized pattern search (GPS) algorithms of this sort in order to keep function evaluations relatively far apart until convergence is approached. The current leading (Cartesian-based) MADS algorithm, OrthoMADS, is extended naturally to the present lattice-based setting by restricting the new poll points to lie on the shell of lattice points that lie k hops from the current CMP at the k'th level of mesh refinement. In such shells, there is a rapidly-growing set of points to select the poll points from in the lattice-based setting as k is increased (dubbed

the ‘coordination sequence’), thus leading to poll sets with high angular *and* radial uniformity. A novel mesh coarsening heuristic is also introduced which makes maximum use of the most recent effective polling direction while keeping the underlying mesh appropriately coarse. Numerical tests demonstrate conclusively that the convergence of the resulting Λ-MADS algorithm is significantly faster than previous MADS implementations, thus making improved progress towards the minimum when only a limited number of function evaluations can be afforded. As with other MADS variants, the possible polling directions ultimately become dense on the unit hypersphere as the lattice is refined, thus preserving the guaranteed convergence characteristics of the MADS class of algorithms as the number of function evaluations ultimately becomes large.

4.2 Background

Practical applications in engineering, science, finance, business, and elsewhere often call for efficient derivative-free algorithms for the optimization of expensive non-smooth functions over a constrained space of n parameters. The field of derivative-free optimization has a long and rich history which includes the development of downhill simplex algorithms, genetic algorithms, and simulated annealing algorithms. The most computationally efficient family of derivative-free optimization algorithms available today, known as *generalized pattern search* (GPS) methods, leverage an underlying mesh to coordinate the exploration of parameter space. The fundamental purpose of this underlying mesh is to keep function evaluations relatively far apart until convergence is approached. All GPS implementations developed by other groups, that we have seen to date, use Cartesian grids to coordinate the exploration of parameter space.

Lattice theory (which builds on the closely-related subjects of n-dimensional sphere packings and error-correcting codes) provides a natural alternative to Cartesian grids for the discretization of parameter space. Conway & Sloane (1998) provides a comprehensive mathematical reference on many important elements of lattice theory; the succinct up-to-date review of this subject in Chapter 2 lays out out essentially everything that is needed to apply this otherwise somewhat abstruse subject in practical applications. The standard measures of lattice uniformity (described in Conway & Sloane

1998 and summarized in Chapter 2) are

- the *packing density*, Δ [that is, the percentage of the domain contained within the spheres when identical spheres with the largest radius possible such that the spheres do not overlap[1] are centered at each lattice point],
- the *covering thickness*, Θ [that is, the average number of spheres that contain any point in the domain when identical spheres with the smallest radius possible such that the every point in the domain is contained within at least one sphere[2] are centered at each lattice point],
- an appropriately-normalized measure of the *mean-squared quantization error per dimension*, G, and
- the *kissing number*, τ [that it, the number of nearest neighbors of each lattice point].

By all four of these standard measures, Cartesian grids become highly nonuniform as the dimension n of the parameter space under consideration is increased; for example, in $n = 24$ dimensions, the Cartesian grid, $\mathbb{Z}^{24}$, is characterized by $\Delta = 1.150e-10$, $\Theta = 4200263$, $G = 0.08333$, and $\tau = 196560$, whereas the Leech lattice,Λ_{24}, is characterized by $\Delta = 0.001930$, $\Theta = 7.904$, $G = 0.06577$, and $\tau = 48$.

A series of highly (in most dimensions, maximally) dense lattices, referred to as the *laminated* lattices and denoted Λ_n, may be constructed in dimensions $n = 2$ to 23 by appropriately restricting the remarkable Leech lattice mentioned above to successively lower and lower dimensions. For $n = 2$ to 8, the resulting lattices are equivalent, respectively, to the so-called root lattices A_2, D_3, D_4, D_5, E_6, E_7, and E_8, each of which have fairly simple constructions and associated quantization algorithms, as reviewed in Chapter 2; some of the salient properties of these lattices are compared with the corresponding Cartesian grids in Table 4.1, the $\mathbb{Z}^2$ and Λ_2 lattices are visualized in Figure 4.1, and the $\mathbb{Z}^3$ and Λ_3 lattices are visualized in Figure 4.2. A primary focus of our research program is to investigate how such highly uniform n-dimensional lattices may be used to accelerate GPS algorithms[3].

[1]The radius of these nonoverlapping spheres, called the *packing radius*, is usually denoted ρ.

[2]The radius of these overlapping spheres that cover the domain, called the *covering radius*, is usually denoted R.

[3]Note that Conway & Sloane (1998, p. 12) state: "A related application that has not yet received much

Table 4.1: Augmenting the data available in Table 2.2, presented here are the number of available points to select the poll set from at the k'th level of grid refinement; 'L/O:' denotes the LTMADS or OrthoMADS contexts, '$\mathbb{Z}$:' denotes the $\mathbb{Z}$-MADS context, and 'Λ:'denotes the Λ-MADS context

lattice	Available points to select the poll set from as the grid is refined.
$D_2 \cong \mathbb{Z}^2$	L/O: 8, 16, 32, 64, 128, ... (Figs 4.1a, 4.3, and 4.4) $\mathbb{Z}$: 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, ... (Figs 4.1b, 4.5)
$A_2 \cong \Lambda_2$	Λ: 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, ... (Figs 4.1c, 4.6)
$\mathbb{Z}^3$	L/O: 26, 98, 386, 1538, 6146, ... (see Figure 4.2a) $\mathbb{Z}$: 6, 18, 38, 66, 102, 146, 198, 258, 326, 402, ... (Fig. 4.2b)
$D_3 \cong A_3 \cong \Lambda_3$	Λ: 12, 42, 92, 162, 252, 362, 492, 642, 812, 1002, ... (Fig. 4.2c)
$\mathbb{Z}^4$	L/O: 80, 544, 4160, 32896, 262400, ... $\mathbb{Z}$: 8, 32, 88, 192, 360, 608, 952, 1408, 1992, 2720, ...
$D_4 \cong \Lambda_4$	Λ: 24, 144, 456, 1056, 2040, 3504, 5544, 8256, 11736, 16080, ...
$\mathbb{Z}^5$	L/O: 242, 2882, 42242, 660482, 10506242, ... $\mathbb{Z}$: 10, 50, 170, 450, 1002, 1970, 3530, 5890, 9290, 14002, ...
$D_5 \cong \Lambda_5$	Λ: 40, 370, 1640, 4930, 11752, 24050, 44200, 75010, 119720, 182002, ...
$\mathbb{Z}^6$	L/O: 728, 14896, 413792, 12746944, 403964288, ... $\mathbb{Z}$: 12, 72, 292, 912, 2364, 5336, 10836, 20256, 35436, 58728, ...
$E_6 \cong \Lambda_6$	Λ: 72, 1062, 6696, 26316, 77688, 189810, 405720, 785304, 1408104, 2376126, ...
$\mathbb{Z}^7$	L/O: 2186, 75938, 3959426, 239479298, 15105828866, ... $\mathbb{Z}$: 14, 98, 462, 1666, 4942, 12642, 28814, 59906, 115598, 209762, ...
$E_7 \cong \Lambda_7$	Λ: 126, 2898, 25886, 133506, 490014, 1433810, 3573054, 7902594, 15942206, 29896146, ...
$\mathbb{Z}^8$	L/O: 6560, 384064, 37281920, 4412866816, 553517580800, ... $\mathbb{Z}$: 16, 128, 688, 2816, 9424, 27008, 68464, 157184, 332688, 658048, ...
$E_8 \cong \Lambda_8$	Λ: 240, 9120, 121680, 864960, 4113840, 14905440, 44480400, 114879360, 265422960, 561403680, ...

4.2.1 Successive polling (SP)

The simplest prototype GPS algorithm, referred to here as *successive polling* (SP), starts from a candidate minimum point (CMP) on a given mesh and polls (that is, checks) the value of the function at a set of nearest-neighbor mesh points which positively span[4] the feasible neighborhood of the CMP. If a function value lower than that of the CMP is located during the poll, the new best point is defined as the new CMP, and the process repeated; if the poll fails to find a point with a better function value, then the mesh is refined by some integer factor[5], so that the function evaluations on the coarser mesh coincide with points on the refined mesh (and may thus be reused efficiently as the iterations proceed on successively refined meshes), and the process repeated until convergence.

Unfortunately, the prototype SP algorithm described above is convergent (albeit to a local minimum) only if the parameter space being explored is unconstrained and the function being optimized is continuously differentiable[6]; that is, if the function being optimized is sufficiently smooth that, after a sufficient number of grid refinements, the function is locally flat enough that, if the CMP is not yet at a minimum, one of the poll points (which, again, are distributed over a set of directions that positively span the neighborhood of the CMP) is guaranteed to have an improved function value, below that of the CMP. For general nonsmooth functions, for functions that are only piecewise differentiable[7], or even for continuously differentiable functions with hard constraints on the feasible domain in parameter space, the SP algorithm is not always convergent, as the finite number of poll directions available might miss the feasible descent directions around the CMP altogether, regardless of the level of grid refinement. Indeed, in the

---

attention is the use of these packings for solving *n*-dimensional search or approximation problems"; this is exactly the focus of this research program.

[4] A set of lattice points is said to *positively span* the feasible neighborhood of the CMP if any point in the feasible neighborhood of the CMP may be reached by a linear combination with *non-negative coefficients* of the vectors from the CMP to the poll points.

[5] Typically, a factor of two is used, in order to keep the refinement of the mesh as slow as possible as the iteration proceeds.

[6] A function is said to be *continuously differentiable* if its derivative is (a) defined everywhere, and (b) continous.

[7] An example of a *piecewise differentiable* function is one with a cusp (akin to the hard chine along the bottom centerline of the hull of many high-speed boats), with the function being continuously differentiable on either side of the cusp.

constrained case, if the CMP is on the constraint boundary, then in most cases the feasible poll points do *not* positively span the feasible neighborhood of the CMP regardless of the level of grid refinement; this is a key challenge that the poll steps in the MADS class of algorithms, discussed further below, are specifically designed to address.

4.2.2 SMF and LABDOGS

The *surrogate management framework* (SMF) of Booker et al. (1999) is a generalization of the SP method described above that alternates between a SP-type 'poll' step, and 'search' step which cleverly leverages a Kriging-based interpolation of all existing function evaluations in order to identify promising and relatively unexplored regions of parameter space. The work presented in Chapter 3 extends the SMF to incorporate lattices, amongst other significant improvements[8], in a manner intended to make maximal use of each and every function evaluation, which are assumed to be expensive, during the optimization process. The resulting *lattice-based derivative-free optimization via global surrogates* (LABDOGS) algorithm shows a significant improvement in the rate of convergence over the original SMF algorithm.

When used appropriately, the search step of the SMF and LABDOGS algorithms can in fact be used to assure *global* convergence, even when the function being optimized is nonsmooth and/or the parameter space being considered is constrained, despite the fact that the SP-type poll step of the SMF and LABDOGS algorithms, taken on their own, don't even establish local convergence for nonsmooth or constrained functions, as discussed above. That is, the search step of the SMF and LABDOGS algorithms can be designed such that, as the number of function evaluations of the algorithm gets large, the function evaluations ultimately become dense over parameter space, thereby ensuring global convergence (for further discussion, see Torczon 1997, Booker et al. 1999, Jones 2001, and Belitz & Bewley 2011).

4.2.3 Mesh Adaptive Direct Search (LTMADS & OrthoMADS)

Mesh Adaptive Direct Search (MADS) algorithms are an alternative class of GPS

[8]Most notably, a markedly improved search function, as suggested by Jones (2001).

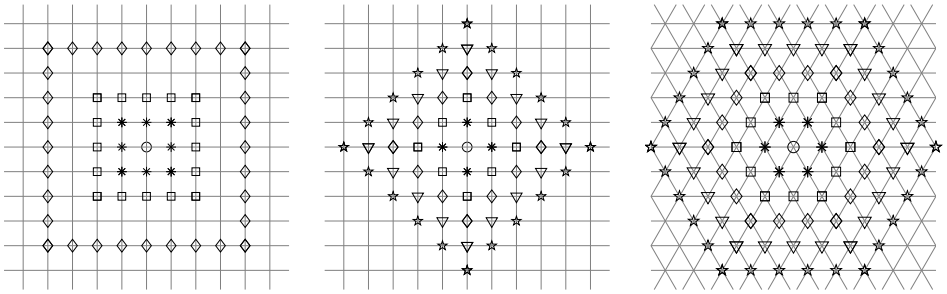

Figure 4.1: The (a, b) $\mathbb{Z}^2$ and (c) Λ_2 lattices, indicating (a) the first three shells of potential poll points on $\mathbb{Z}^2$ used in the LTMADS and OrthoMADS formulations, and (b, c) the first five shells of potential poll points used, respectively, in the $\mathbb{Z}$-MADS and Λ-MADS formulations. The number of points in all three sets of shells (arranged, respectively, in squares, diamonds, and hexagons around the CMP) are listed in Table 4.1.

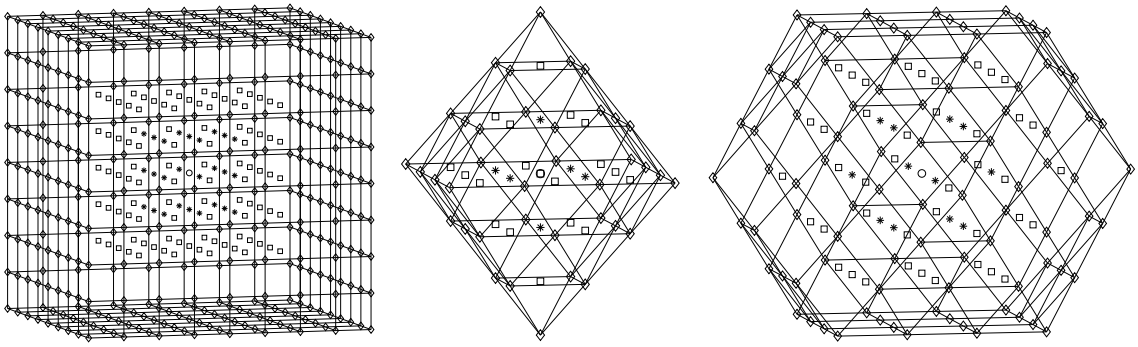

Figure 4.2: The (a, b) $\mathbb{Z}^3$ and (c) Λ_3 lattices, indicating (a) the first three shells of potential poll points on $\mathbb{Z}^3$ used in the LTMADS and OrthoMADS formulations, and (b, c) the first three shells of potential poll points used, respectively, in the $\mathbb{Z}$-MADS and Λ-MADS formulations. The number of points in all three sets of shells (arranged, respectively, in cubes, octahedra, and cuboctahedra around the CMP) are listed in Table 4.1.

methods designed to overcome the fundamental convergence shortcoming of the polling algorithm used in the prototype SP method (and built upon in the SMF and LABDOGS methods), as described above. They accomplish this by increasing (without bound) the number of directions around current CMP that may be polled as the grid is refined; as the number of grid refinements performed increases, the possible polling directions ultimately become dense over the feasible neighborhood of the CMP. This is achieved in the MADS setting, in general, by selecting the poll points from a shell[9] of non-nearest-neighbor mesh points around the CMP.

Existing variants of MADS include LTMADS (Abramson, Audet, & Dennis 2005; for a graphical depiction, see Figure 4.3), which is quite popular for difficult numerical optimization (see, e.g., Marsden et al., 2011), and OrthoMADS (Audet & Dennis 2008; for a graphical depiction, see Figure 4.4), the latter of which essentially supercedes the former. Both LTMADS and OrthoMADS are based on an underlying Cartesian grid $\mathbb{Z}^n$, with LTMADS based on *minimal* positive bases, with $n+1$ vectors around the CMP, and OrthoMADS based on *maximal* positive bases, with $2n$ vectors around the CMP. In both the LTMADS and OrthoMADS algorithms, the underlying grid is refined by a factor of *four* upon each refinement of the grid[10], whereas the shell of points from which the next poll set is to be selcted lie on a hypercube around the CMP whose width is reduced only by a factor of *two* upon each refinement of the grid. That is, the set of potential poll points around the CMP in the LTMADS and OrthoMADS formulations is the set of points on the $\mathbb{Z}^n$ grid of L_∞ norm 2^k (see Figures 4.1a and 4.2a), scaled down by a factor of $1/4^k$, where $k = 0, 1, 2, \ldots$ is the number of grid refinements performed thus far; the LTMADS algorithm will select $n+1$ of these points to poll (see Figure 4.3), whereas the OrthoMADS algorithm will select $2n$ of these points to poll (see Figure 4.4). Thus, as the underlying Cartesian grid is successively refined in LTMADS and OrthoMADS, the shell of points from which the poll is selected contains successively more and more points, ultimately increasing in number by a factor of $\sim 2^{n-1}$ upon each refinement of the $\mathbb{Z}^n$ grid (see Table 4.1). Given an appropriate

[9] We use the word '*shell*' in this work to denote the surface of the region given by the convex hull of the specified points.

[10] That is, after just five grid refinements, the refined grid that has less than $1/1000$ of the original grid spacing in every coordinate direction.

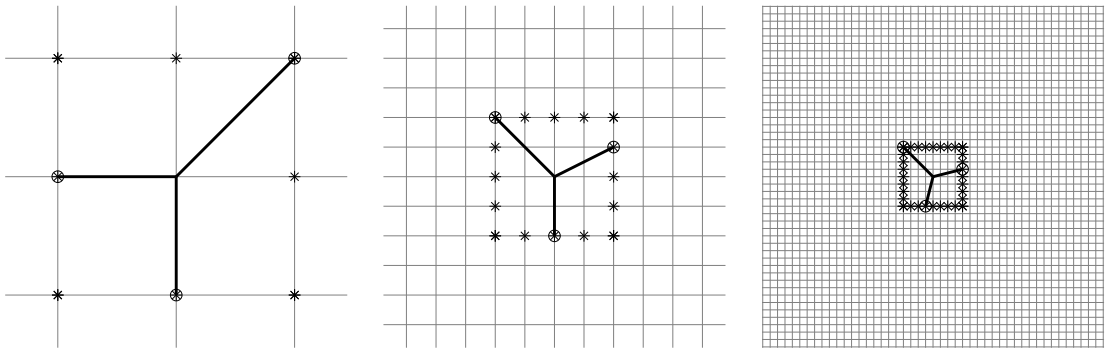

Figure 4.3: The underlying Cartesian grid $\mathbb{Z}^2$ (——) and two successive factor-of-four refinements of this grid (from left to right) in the $n = 2$ LTMADS algorithm. Given a CMP at the center of each subfigure, the shell of points from which the poll points are selected are marked ($*$), and a representative poll set is indicated ($\circ$); this poll set forms a minimal positive basis (——), with $n+1$ vectors around the CMP.

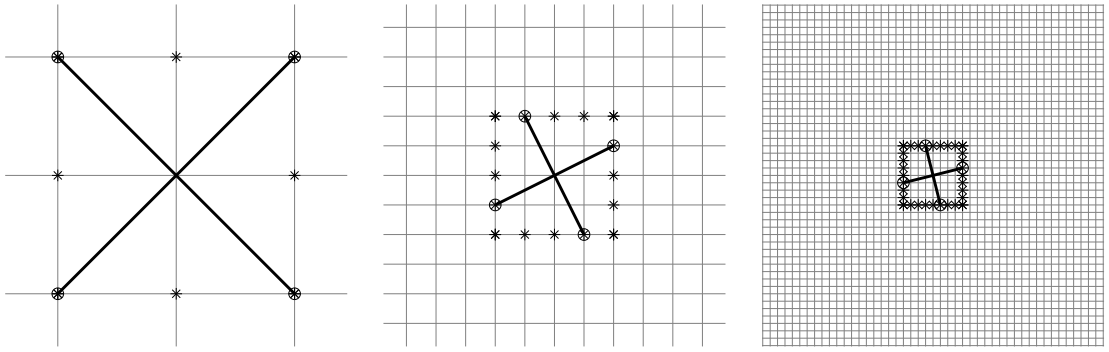

Figure 4.4: The underlying Cartesian grid $\mathbb{Z}^2$ (——) and two successive factor-of-four refinements of this grid in the $n = 2$ OrthoMADS algorithm (cf. Figure 4.3). The shell of points from which the poll points are selected are marked ($*$), and a representative poll set is indicated ($\circ$); this poll set forms an (orthogonal) maximal positive basis (——), with $2n$ vectors around the CMP.

scheme for selecting the poll points to actually use from this shell of possible poll points around the CMP, convergence (albeit, to local minima) of the MADS algorithm may thus be established (see Abramson, Audet, & Dennis 2005 and Audet & Dennis 2008) even when the function being optimized is nonsmooth, and/or the parameter space being considered is constrained.

LTMADS selects the first 'seed' vector of the poll set using a pseudo-random

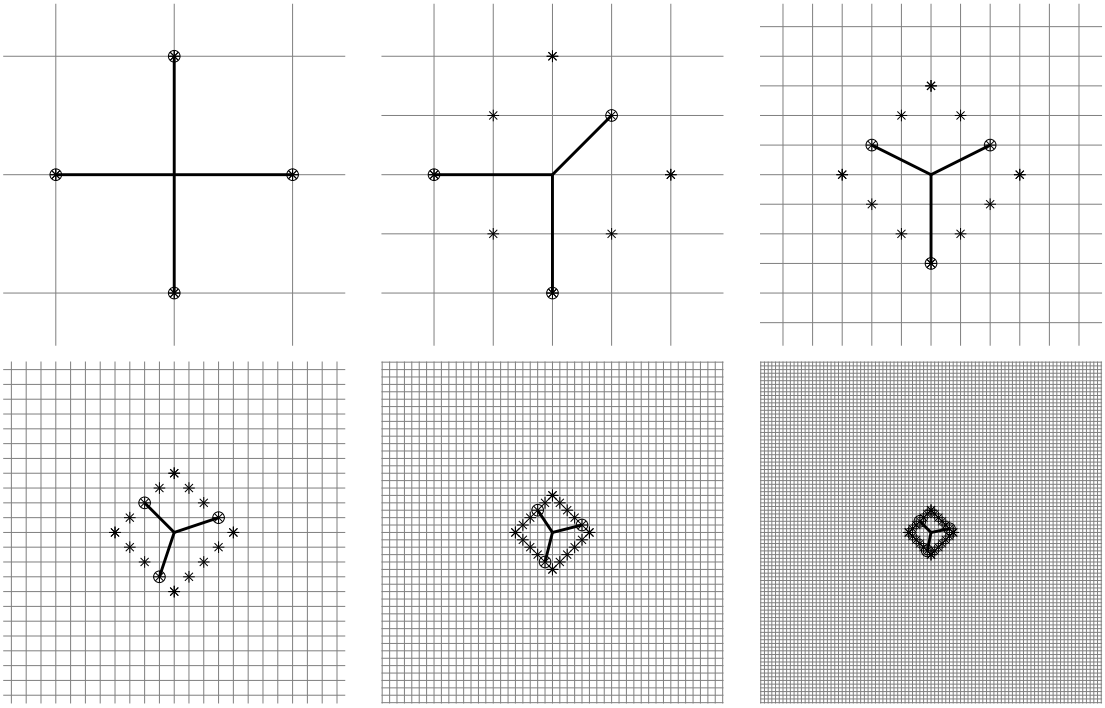

Figure 4.5: The underlying Cartesian grid $\mathbb{Z}^2$ (——) and five successive factor-of-two refinements of this grid in the $n = 2$ $\mathbb{Z}$-MADS algorithm (cf. Figures 4.3 and 4.4). The set of points from which the poll points are selected are marked ($*$), and a representative poll set is indicated ($\circ$); this poll set forms a maximal positive basis around the CMP on the original grid and a minimal positive basis around the CMP on the others (▬).

algorithm, then builds a minimal positive basis via a stochastic *lower triangular* construction (thus motivating the algorithm name); for details, see Abramson, Audet, & Dennis (2005). As illustrated in in Figure 4.3, the radial and angular uniformity of the poll sets generated by the LTMADS algorithm can both be poor; the poll set shown in in Figure 4.3a has one poll vector that is $\sqrt{n}$ longer than the others, and the angles between the poll vectors vary from 90° to 135°.

OrthoMADS, in contrast, selects the first 'seed' vector of the poll set using a (low discrepancy) 'quasi-random' Halton sequence, builds up set of $n-1$ directions that are *orthogonal* to this seed (thus motivating the algorithm name) via a Householder-based QR algorithm, then finds the $2n$ points amongst the (hypercube-shaped) shell of potential poll points that are closest to these directions and their opposites; for details, see Audet & Dennis (2008). As illustrated in in Figure 4.4, the radial and angular uniformity of the

poll sets generated by the OrthoMADS algorithm for $n = 2$ are perfect. Unfortunately, for $n > 2$, the radial and angular uniformity of the OrthoMADS poll sets can, again, both be poor. Consider, e.g., the case with $n = 3$ and a first seed vector of the poll set oriented torwards one of the corners of the cube; it is clearly not possible to select the remaining five poll points to provide both good radial uniformity[11] and good angular uniformity in this case. Note also that an OrthoMADS poll require $2n$ function evaluations to complete, rather than the $n+1$ function evaluations required to complete a poll on a minimum positive basis, such as that used by LTMADS; for larger values of n, this fact alone results in about a factor of 2 reduction in the rate of convergence.

4.2.4 Slowing the mesh refinement of Cartesian-based MADS algorithms ($\mathbb{Z}$-MADS)

Before we discuss shifting the MADS algorithm onto a more uniform lattice, we first note that the factor-of-four method of successive refinement, as described in the previous section and illustrated in Figures 4.3 and 4.4, is not the only choice for a MADS-type algorithm on a Cartesian grid. As illustrated in Figure 4.5, the Cartesian grid may instead be refined only by a factor of *two* whenever a poll step fails; this helps to slow the refinement of the underlying mesh as the iterations proceed, thus respecting the overall GPS objective of keeping function evaluations relatively far apart until convergence is approached. As the Cartesian grid is refined in this modified approach, which we will call $\mathbb{Z}$-MADS, the shell of points around the CMP from which the poll points are selected is increased one 'hop' at a time (see Figures 4.1b & 4.2b). Thus, as the underlying Cartesian grid is successively refined in $\mathbb{Z}$-MADS, the shell of points from which the poll is selected ultimately decreases in width by a factor of ~ 2 upon each refinement of the grid. Further, as the underlying Cartesian grid is successively refined, the shell of points from which the poll is selected again contains successively more and more points; the available points to select the poll set from in this case is the number of points k hops from the origin on $\mathbb{Z}^n$ for $k = 1, 2, 3, \ldots$ (that is, the *coordination sequence* of $\mathbb{Z}^n$, as listed in Table 4.1). Noting the discussion at the end of the previous section, at each poll step, the $\mathbb{Z}$-MADS algorithm selects $n+1$ of these points

[11] The radial nonuniformity of this approach is quantified in §4.3.

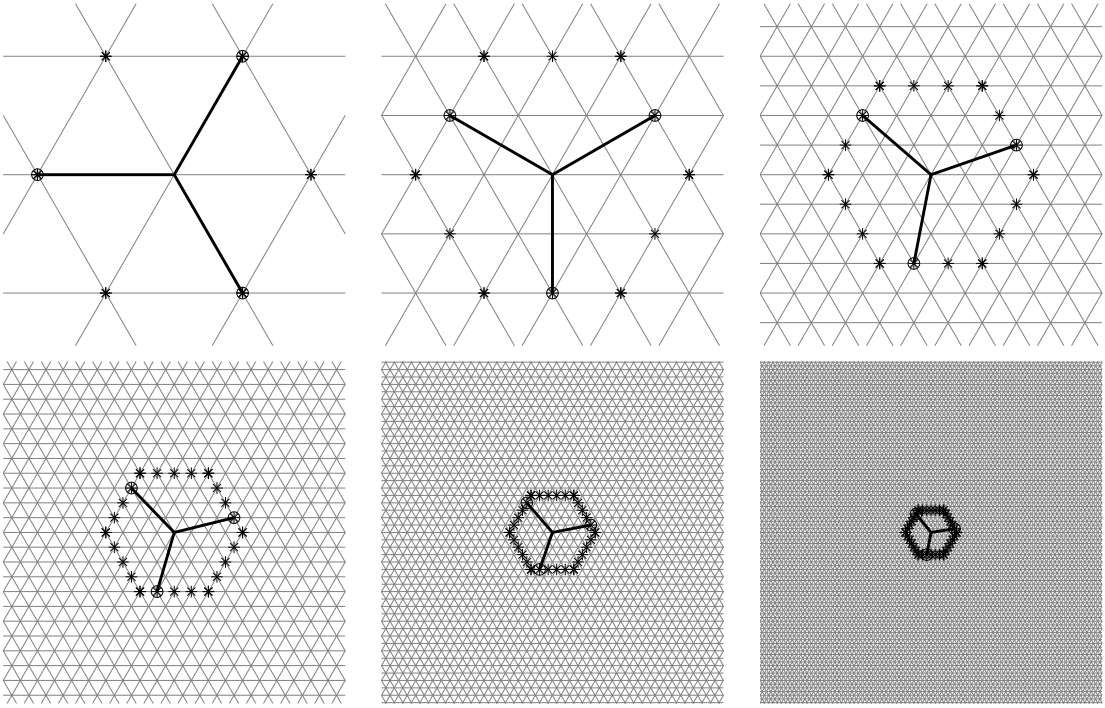

Figure 4.6: The underlying Λ_2 lattice (——) and five successive factor-of-two refinements of this lattice in the $n = 2$ Λ-MADS algorithm (cf. Figures 4.3, 4.4, and 4.5). The set of points from which the poll points are selected are marked ($*$), and a representative poll set is indicated ($\circ$); this poll set forms a minimal positive basis around the CMP (——).

to poll.

4.2.5 An overview of Lattice-based MADS (Λ-MADS)

The present work demonstrates how uniform lattices of the Λ_n family may be used to significantly accelerate the convergence of the MADS class of algorithms in order to solve the constrained nonsmooth optimization problem $\operatorname{argmin}\{f(\mathbf{x}) : \mathbf{x} \in \Omega\}$ where $\Omega \subset \mathbb{R}^n$. The function $f(\mathbf{x})$ to be minimized is treated in this setting as a 'black box' for which derivative information is perhaps impossible to derive and, even if it can be derived, is possibly poorly behaved due to the potentially nonsmooth nature of the function of interest. The resulting optimization algorithm, dubbed Λ-MADS (for a graphical depiction, see Figure 4.6), follows naturally from the $\mathbb{Z}$-MADS algorithm

described above, with the exploration of parameter coordinated by the laminated lattices Λ_n rather than the Cartesian grid $\mathbb{Z}^n$. As discussed in Conway & Sloane (1997) and Baake & Grimm (1997), closed-form expressions of the coordination sequences of Λ_2 through Λ_8 (that is, the number of potential polling points in each shell used by the Λ-MADS algorithm) are given by the coefficients of the series expansions at $x = 0$ of the following expressions: $S_{\Lambda_2}(x) = (1+4x+x^2)/(1-x)^2$,
$S_{\Lambda_3}(x) = (1+9x+9x^2+x^3)/(1-x)^3$,
$S_{\Lambda_4}(x) = (1+20x+54x^2+20x^3+x^4)/(1-x)^4$,
$S_{\Lambda_5}(x) = (1+35x+180x^2+180x^3+35x^4+x^5)/(1-x)^5$,
$S_{\Lambda_6}(x) = (1+66x+645x^2+1384x^3+645x^4+66x^5+x^6)/(1-x)^6$,
$S_{\Lambda_7}(x) = (1+119x+2037x^2+8211x^3+8787x^4+2037x^5+119x^6+x^7)/(1-x)^7$,
$S_{\Lambda_8}(x) = (1+232x+7228x^2+55384x^3+133510x^4+107224x^5+24508x^6+232x^7+x^8)/(1-x)^8$.

Such series expansions are easily calculated in Mathematica or Wolfram|Alpha or similar ; the first 10 terms of each of these series are listed in Table 4.1.

The resulting Λ-MADS algorithm is quite straightforward to use, though significant care must be exercised on several subtle issues in its implementation in order to ensure the maximum rate of convergence of the resulting algorithm; after exploring a bit further the some geometrical considerations of this formulation in §4.3, these implementation issues are addressed at length in §4.4. In §4.5, we attempt to quantify the impact of each of the individual implementation issues discussed here and in §4.4 in focused numerical experiments; we then verify that the final Λ-MADS algorithm converges significantly faster than the previous OrthoMADS algorithm on some representative test problems, and provide some concluding remarks.

4.3 Geometrical considerations

We now consider further some relevant n-dimensional geometrical issues related to this optimization framework. We are specifically interested in n-dimensional *convex polytopes*, that is, in n-dimensional convex objects with flat sides, more commonly called *polygons* in $n = 2$ dimensions, *polyhedrons* in $n = 3$ dimensions, and *polychorons*

in $n = 4$ dimensions (a good reference on this general subject area is Grünbaum 2002).

The *Voronoi cell* of a lattice is the set of all points that are as close to the origin as they are to any other lattice point; stated another way, the Voronoi cell contains exactly those points that quantize to the origin (or, shifting the Voronoi cell appropriately, to any other lattice point) when performing lattice quantization. The dual of any convex polytope may be formed by the process of *polar reciprocation* (Grünbaum 2002). The dual of the Voronoi cell is called the *Delaunay cell*.

On the Cartesian lattice and the root lattices A_n, D_n, E_6, E_7, and E_8, the Voronoi cells are established solely by the locations of the nearest neighbors to the origin. As discussed further in Chapter 21 of Conway & Sloane 1998, defining τ as the kissing number of the corresponding n-dimensional lattice, the Voronoi cells of these lattices may be constructed by the union of τ identical (but rotated) *fundamental simplices*, each of which has the origin and n other points as vertices (identified precisely in Figures 21.6, 21.7, and 21.8 of Conway & Sloane 1998). The $(n-1)$-dimensional face of each fundamental simplex that is opposite to the origin forms a perpendicular bisector of the line segment between the origin and each of the nearest neighbors of the origin on the corresponding lattice; the Voronoi cell is then the convex n-dimensional region contained by all τ of these $(n-1)$-dimensional faces. So defined, the Voronoi cell of the $A_2 \cong \Lambda_2$ lattice is a *hexagon* (with $\tau = 6$ one-dimensional faces, a.k.a. edges), the Voronoi cell of the $D_3 \cong A_3 \cong \Lambda_3$ lattice is a *rhombic dodecahedron* (with $\tau = 12$ two-dimensional faces), and the Voronoi cell of the $D_4 \cong \Lambda_4$ lattice is a *24-cell* (a.k.a. *icositetrachoron*, with $\tau = 24$ three-dimensional faces); the Voronoi cells of Λ_5 through Λ_8 are less commonly known structures, but are constructed in the same fashion. The Delaunay cells of these lattices (that is, the duals of the corresponding Voronoi cells) are each simply the convex hull of the nearest neighbors of the origin; thus, the Delaunay cell of the Λ_2 lattice is also a *hexagon* (rotated 30° from the corresponding Voronoi cell), the Delaunay cell of the Λ_3 lattice is a *cuboctahedron*, and the Delaunay cell of the Λ_4 lattice is also a *24-cell* (again, rotated).

As discussed previously, the LTMADS and OrthoMADS formulations build out shells of potential polling points in the shapes of hypercubes (see Figures 4.1a and 4.2a), which are precisely the shape of the Voronoi cells of the corresponding Cartesian lattice

$\mathbb{Z}^n$. In $n = 2$ to 8 dimensions, a hypercube goes by the following names: *square*, *cube*, *tesseract*, *penteract*, *hexeract*, *hepteract*, and *octeract*.

The $\mathbb{Z}$-MADS formulation, in contrast, builds out shells of potential polling points a given number of hops from the CMP on the $\mathbb{Z}^n$ lattice (see Figures 4.1b and 4.2b). These shells are precisely in the shapes of the convex hulls of the nearest neighbors of the origin (that is, of the corresponding Delaunay cells, or the duals of the corresponding Voronoi cells); note specifically that the duals of hypercubes are known as *cross polytopes*. In $n = 2$ to 8 dimensions, a cross polytope goes by the following names: *square*[12], *octahedron*, 16-*cell*, *pentacross*, *hexacross*, *heptacross*, and *octacross*.

Similarly, the Λ-MADS formulation builds out sets of potential polling points a given number of hops from the CMP on the Λ_n lattice (see Figures 4.1c and 4.2c). These shells are precisely in the shapes of the corresponding Delaunay cells which, for $n = 2$ to 8 dimensions, are simply the convex hulls of the lattice points that are nearest neighbors of the origin in the corresponding Λ_n lattice, as described above.

The resulting shell shapes in the LTMADS/OrthoMADS, $\mathbb{Z}$-MADS, and Λ-MADS formulations are summarized in Table 2.2. The radial nonuniformity of each of these shells is defined here as the maximal radius of the shell (at a vertex) divided by the minimal radius of the shell (at the center of a face), and quantifies the maximum radial nonuniformity possible in the corresponding poll sets. Remarkably, due to the polar reciprocation process mentioned previously, which relates a convex polytope and it's dual, the radial nonuniformity of a Voronoi cell and the the radial nonuniformity of the corresponding Delaunay cell of a lattice are, in fact, equal. Using the notation introduced previously, they are both given by the covering radius divided by the packing radius [that is, by R/ρ] of the lattice, and may thus also be written as the n'th root of the covering thickness divided by the n'th root of the packing density [that is, by $(\Theta/\Delta)^{1/n}$] of the lattice[13].

[12]Since in the present case the Delaunay cell is rotated 45° from the corresponding Voronoi cell, the cross polytope forming the Delaunay cell in the $n = 2$ case is perhaps better identified as a '*diamond*'.

[13]Recall that both the covering thickness Θ and the packing density Δ of the lattices of interest in this work are listed in Table 4.1; thus, the radial nonuniformity values presented in Table 4.2 may be derived directly from the Θ and Δ values presented in Table 4.1.

Table 4.2: Radial nonuniformity of the shell of potenial poll points in the LTMADS/OrthoMADS, $\mathbb{Z}$-MADS, and Λ-MADS formulations, as a function of the dimension n.

	LTMADS/OrthoMADS		$\mathbb{Z}$-MADS		Λ-MADS	
n	shell shape	radial nonuniformity	shell shape	radial nonuniformity	shell shape	radial nonuniformity
2	square	$\sqrt{2} \approx 1.41$	diamond	$\sqrt{2} \approx 1.41$	hexagon	$\sqrt{4/3} \approx 1.16$
3	cube	$\sqrt{3} \approx 1.73$	octahedron	$\sqrt{3} \approx 1.73$	cuboctaheron	$\sqrt{2} \approx 1.41$
4	tesseract	$\sqrt{4} = 2.00$	16-cell	$\sqrt{4} = 2.00$	24-cell	$\sqrt{2} \approx 1.41$
5	penteract	$\sqrt{5} \approx 2.24$	pentacross	$\sqrt{5} \approx 2.24$	(see text)	$\sqrt{5/2} \approx 1.58$
6	hexeract	$\sqrt{6} \approx 2.45$	hexacross	$\sqrt{6} \approx 2.45$	(see text)	$\sqrt{8/3} \approx 1.63$
7	hepteract	$\sqrt{7} \approx 2.65$	heptacross	$\sqrt{7} \approx 2.65$	(see text)	$\sqrt{3} \approx 1.73$
8	octeract	$\sqrt{8} \approx 2.83$	octacross	$\sqrt{8} \approx 2.83$	(see text)	$\sqrt{2} \approx 1.41$

It may finally be observed that, in all dimensions, the shells of potential poll points in the LTMADS/OrthoMADS and $\mathbb{Z}$-MADS formulations are characterized by significantly more severe radial nonuniformity than the shell of potential poll points in the corresponding Λ-MADS formulation, with the differences becoming especially pronounced as n is increased, as quantified in Table 4.2. This observation, in addition to the significantly improved spatial uniformity of the Λ_n lattices as compared with the $\mathbb{Z}^n$ grids used previously (apparently, by default) for the coordination of MADS algorithms, are two key motivations for the present investigation.

4.4 Issues affecting the implementation and the speed of convergence of the Λ-MADS algorithm

The basic idea of the Λ-MADS algorithm has already been laid out. To recap: starting with an initial, relatively coarse[14] lattice with nearest neighbors spaced Δ_0 apart, and starting from an initial feasible candidate minimum point (CMP) on this lattice, a set of $n+1$ points which are nearest neighbors to the CMP on the lattice are selected in such a way as to *positively span* (that is, to linearly span with non-negative coefficients) the neighborhood of the CMP. The value of the function is then *polled* (that is, checked) on these points. If a poll point is found with a lower function value than that of the CMP, then this new lattice point is defined as the new CMP, and the process repeated; if not, then the lattice is refined by factor of two, a new poll set (randomly reoriented) is chosen on the refined lattice (from a shell of potential poll points containing all lattice points that are $k+1$ hops from the CMP, where k is the number of lattice refinements performed thus far), and the process repeated until convergence. There are a number of subtle issues that must be addressed in order to specify this algorithm completely, and to endow it with the maximum possible efficiency. These issues are now addressed.

[14] An initial grid spacing of about four to eight gridpoints from one edge of the feasible domain to the other in each parameter direction has proven to be effective in our numerical experiments performed to date. Note also that, in general, the scaling of each parameter in the optimization problem of interest is found to have a significant effect on the rate of convergence of a GPS algorithm; the most effective scalings are those in which, on average, the function of interest varies at approximately the same rate in each coordinate direction; this provides a general goal to strive for when setting up an optimization problem for solution via a GPS algorithm.

4.4.1 Moving around on, and quantizing to, the laminated lattices Λ_n

The *theory* of *n*-dimensional lattices is quite sophisticated (see Conway & Sloane 1998); however, the *practical use* of *n*-dimensional lattices is entirely straightforward, as discussed in Chapter 2. Once the enumeration and quantization algorithms for any given lattice are in place, as discussed below, the lattice may be used in the present application in a straightforward manner.

Any real lattice is defined simply by all *integer linear combinations*[15] of the columns of an appropriate basis matrix *B*. Basis matrices for the seven laminated lattices considered in this chapter, Λ_2 through Λ_8, are given by

$$B_{\Lambda_2} = \begin{pmatrix} -1 & \\ 1 & -1 \\ & 1 \end{pmatrix}, \quad B_{\Lambda_3} = \begin{pmatrix} -1 & 1 & \\ -1 & -1 & 1 \\ & & -1 \end{pmatrix}, \quad B_{\Lambda_4} = \begin{pmatrix} -1 & 1 & & \\ -1 & -1 & 1 & \\ & & -1 & 1 \\ & & & -1 \end{pmatrix},$$

$$B_{\Lambda_5} = \begin{pmatrix} -1 & 1 & & & \\ -1 & -1 & 1 & & \\ & & -1 & 1 & \\ & & & -1 & 1 \\ & & & & -1 \end{pmatrix}, \quad B_{\Lambda_6} = \begin{pmatrix} & & & & & 1/2 \\ -1 & & & & & 1/2 \\ 1 & -1 & & & & 1/2 \\ & 1 & -1 & & & 1/2 \\ & & 1 & -1 & & -1/2 \\ & & & 1 & -1 & -1/2 \\ & & & & 1 & -1/2 \\ & & & & & -1/2 \end{pmatrix}$$

[15] That is, all linear combinations with integer coefficients.

$$B_{\Lambda_7} = \begin{pmatrix} -1 & & & & & & 1/2 \\ 1 & -1 & & & & & 1/2 \\ & 1 & -1 & & & & 1/2 \\ & & 1 & -1 & & & 1/2 \\ & & & 1 & -1 & & -1/2 \\ & & & & 1 & -1 & -1/2 \\ & & & & & 1 & -1/2 \\ & & & & & & -1/2 \end{pmatrix},$$

$$B_{\Lambda_8} = \begin{pmatrix} 2 & -1 & & & & & & 1/2 \\ & 1 & -1 & & & & & 1/2 \\ & & 1 & -1 & & & & 1/2 \\ & & & 1 & -1 & & & 1/2 \\ & & & & 1 & -1 & & -1/2 \\ & & & & & 1 & -1 & -1/2 \\ & & & & & & 1 & -1/2 \\ & & & & & & & -1/2 \end{pmatrix}.$$

Note that, in the simple representations used above, Λ_2, Λ_6, and Λ_7 are defined on hyperplanes of higher-dimensional spaces; this presents only a relatively minor added complexity when enumerating the lattice points according to these definitions. Several properties of the seven lattices so defined are listed in Table 4.1. Associated with each of these lattices is a straightforward and computationally efficient *quantization* algorithm, described in §1.5, which takes any point in $\mathbb{R}^n$ and computes the closest point on the discrete lattice Λ_n.

Enumerating the nearest neighbors of a lattice

In the computational implementation of the Λ-MADS algorithm, it is numerically tractable and convenient to enumerate explicitly the nearest neighbors of the origin of the lattice. These nearest neighbors may be determined by taking all integer linear combinations of the associated basis vectors, defined above, for integer coefficients ranging from $-m$ to $+m$ (initially taking, say, $m = 2$), and keeping the distinct

lattice points so generated that are closest to the origin; if there are τ such points generated (where τ is listed for each lattice in Table 4.1), then finish, otherwise, increase m by one and try again.

Bypassing the enumeration of subsequent shells of a lattice in the practical Λ-MADS algorithm

For small values of n, it is also numerically tractable to compute the first few shells of neighbors outside of the nearest neighbors, as depicted in Figures 4.1c & 4.2c. These subsequent shells may be created by shifting the nearest-neighbor shell to each point of the outermost shell determined thus far, and keeping track of all of the distinct new lattice points so generated. This method is computationally efficient for shells containing up to a few thousand lattice points.

However, for shells that contain more than a few thousand lattice points (that is, for the outer shells in the higher dimensions n), the direct enumeration procedure described above becomes numerically intractable.

We thus avoid completely the direct enumerations of the shells outside of the nearest-neighbor shell in the practical Λ-MADS algorithm. Instead, we determine the average radius of each target shell of points around the CMP[16], and work directly with the (normalized) desired poll *directions*, scaling these directions by the average radius of the target shell and then quantizing to the nearest lattice point in order to generate the corresponding poll point. For target shells of small radius (that is, at most a few hops from the CMP), this approach returns poll points on the target shell itself, as depicted in Figures 4.1c & 4.2c. For target shells of larger radius, however, this approach returns poll points with, in fact, somewhat improved radial uniformity than is possible when strictly using only points on the target shell itself. This relaxation of the strict use of the shells defined in terms of number of hops from the origin is found to work quite effectively in practice.

[16]Knowing the nearest-neighbor distance at the present level of grid refinement, as well as the radial nonuniformity of the target shell from Table 4.2, the average radius of each target shell can be well approximated quite easily.

4.4.2 Evaluating the poll points: complete polling versus incomplete polling

If a function value lower than that of the CMP is located during the poll step of Λ-MADS, the poll may be terminated immediately, the new best point defined as the new CMP, and the process repeated (a strategy referred to as *incomplete polling*); alternatively, the poll step may be driven all the way to completion, after which the best point found during the polling is identified as the new CMP (a strategy referred to as *complete polling*). In all GPS settings that we have tested to date, our numerical experiments indicate that, on average, incomplete polling is generally the most efficient choice; incomplete polling is thus implemented in Λ-MADS.

4.4.3 Refining the mesh

As mentioned previously and illustrated in Figure 4.6, the lattice is refined only by a factor of two, rather than a factor of four, whenever a poll step fails in the algorithm we propose; this helps to slow the refinement of the underlying mesh as the iterations proceed, thus respecting the overall GPS objective of keeping function evaluations relatively far apart until convergence is approached.

As in $\mathbb{Z}$-MADS, as the lattice is refined in Λ-MADS, the shell of points around the CMP from which the poll points are selected is increased essentially[17] one hop at a time (see Figures 4.1c & 4.2c and Figure 4.6). This shell is much closer to spherical than are the shells of points considered in the LTMADS/OrthoMADS and $\mathbb{Z}$-MADS contexts, as quantified in Table 4.2 of §4.3. As a consequence, the radial uniformity of the Λ-MADS poll sets is substantially better than the radial uniformity of the LT-MADS/OrthoMADS and $\mathbb{Z}$-MADS poll sets.

The available points to select the poll set from as the Λ_n grid is refined is thus given (again, essentially[17]) by the coordination sequence of the corresponding lattice; the Λ-MADS algorithm will select $n+1$ of these points to poll, unless previous function evaluations are available which may be exploited (for further discussion, see §4.4.4). As

[17] As mentioned in §4.4.1, this method is modified slightly in the practical Λ-MADS algorithm for the outer shells.

listed in Table 4.1, the coordination sequence of the Λ_n lattice grows faster than the coordination sequence of the corresponding $\mathbb{Z}^n$ lattice, and thus there are more points to pick from in Λ-MADS than there are in $\mathbb{Z}$-MADS at any given level of mesh refinement[18].

Note that ten factor-of-two grid refinements corresponds to a refined grid that has less than $1/1000$ of the original grid spacing in every coordinate direction. As the dimension of the problem under consideration is increased, this is probably essentially as far as most practical derivative-free optimization problems would ever be taken; the behavior as the number of grid refinements is taken to infinity is, from the perspective of difficult practical problems to be solved with limited computational resources, mostly a mathematical curiosity.

Thus, in addition to a *coarsest* grid spacing to be used by the optimization algorithm (see the first paragraph of §4.4), it is useful in the practical implementation of Λ-MADS to also set a *finest* grid spacing to be used by the optimization algorithm. Note in Table 4.1 that, after about ten factor-of-two grid refinements in the Λ-MADS algorithm, there are a *lot* of points available to select the poll set from. Once on this finest grid, rather than refining the grid even further after each failed poll step, it is practically useful to remain on this finest grid level until all of the potential polling points at this level have, one poll set at a time, been exhaustively checked (or the CPU time allocated to perform the optimization has run out), after which, if all of these poll sets fail to provide a new CMP, the optimization algorithm simply terminates. There is little practical use to refine the grid even further than this, and so doing can actually lead to a substantially reduced overall rate of convergence and an increased sensitivity to numerical precision issues, as the step size gets impractically small when too many grid refinements are performed.

[18]There are in fact many more points available in the LTMADS/OrthoMADS context after a given number of mesh refinements than there are in the Λ-MADS context after the same number of mesh refinements. However, an argument may be made that there is no real "need", from a convergence persepective, for the number of available points in the shells of potential poll points in a MADS-type algorithm to grow so quickly; a MADS algorithm will only evaluate a small subset of the points in any given shell anyway. The fact that the number of points in each successive shell grows without bound is enough to establish convergence of the corresponding MADS algorithm. As far as we can tell, the fact the number of points in the LTMADS/OrthoMADS shells grows extremely quickly (see Table 4.1) does not actually benefit the overall rate of convergence of the practical LTMADS or OrthoMADS algorithms.

4.4.4 Generating new poll sets

Minimizing the number of new function evaluations required in each poll

A significant difference between LTMADS and OrthoMADS, as described previously, is that one uses a minimal positive basis at each poll step, whereas the other uses a maximal positive basis at each poll step. The numerical tests that we have performed to date indicate that, all other things being equal (including the approximate angular and radial uniformity of the respective poll sets), it is usually more efficient computationally to minimize the number of new function evaluations required in each poll step, especially as the dimension n of the problem is increased; thus, when no previous function evaluations are available which may be exploited (for further discussion, see §4.4.4), the use of minimal positive bases is generally preferred. This is not a strong preference however, and it is entirely straightforward to implement poll sets with more than $n+1$ poll points in the Λ-MADS algorithm.

Generating a uniform poll set that positively spans the neighborhood of the CMP leveraging a Thompson algorithm

The flexible algorithm that we use to actually generate poll sets with good angular uniformity in the present work while performing the minimum number of new function evaluations possible in each poll step is derived directly from the method developed in §II.B of Belitz & Bewley (2011) and Chapter 3. In brief, to generate p poll points[19] on the target shell with good angular uniformity from the CMP, we first model p "charged particles" distributed randomly on a sphere with radius given by the average radius of the target shell. A Thompson algorithm is then used to drive this set of particles to an equilibrium configuration on this sphere. The final equilibrium position of these particles is then discretized to the nearest lattice points, as motivated by the third paragraph of §4.4.1. Finally, these discretized points are checked to ensure that they positively span the neighborhood of the CMP, a test for which is given in §II.A of Belitz & Bewley (2011) and Chapter 3. If points so generated do not positively span the

[19] We may initially take $p = n+1$; note that this algorithm is easily and naturally extended in three important ways in §4.4.4, §4.4.4, and §4.4.4.

neighborhood of the CMP, a different random initial distribution of the p particles on the sphere may be tried, and the process repeated; if the process still fails to produce a discretized set of p points that positively span the neighborhood of the CMP, p is incremented by one, and the process repeated until a positively spanning set of poll points is successfully found.

Implementing constraints on the feasible parameter domain

The feasible domain of parameter space over which the optimization is performed might in fact be difficult or impossible to identify and characterize a priori. Thus, the constraints on the feasible domain of parameter space are ignored completely at the stage of selecting which specific points from the target shell are to be polled. If a given poll point proves to be infeasible when it is ultimately evaluated, the corresponding function value is simply set to infinity (or, to an arbitrarily large value), and the poll step is continued. Since interpolating functions are not used by the Λ-MADS algorithm (in contrast with the SMF and LABDOGS algorithms mentioned previously), this simple manner of handling the implementation of constraints is entirely adequate.

Reusing existing function evaluations during each poll step

It is a simple matter to incorporate m existing function evaluations available on or within the target shell in the process described in §4.4.4: "fixed" charged particles are simply assigned to points on the unit sphere corresponding to the existing function evaluations (that is, scaling their distance from the CMP appropriately), and other "free" charged particles are allowed to move to equilibrium positions on the sphere in the manner described in the previous section; the equilibrium positions of these free particles are then discretized to the nearest lattice points to generate the new poll points. By so doing, the number of new function evaluations required to complete a poll step (which, taken together with the existing function evaluations, positively spans the neighborhood of the CMP) can often be reduced significantly.

Reorienting the poll set in a low-discrepency fashion after one or more unsuccessful poll steps, leveraging a Thompson algorithm

If a given poll step in the Λ-MADS algorithm fails to identify a new CMP, after refining the mesh and incrementing the shell containing the possible poll points, the poll set must be reoriented. It is desirable that the orientation of this new poll set explore new directions around the CMP, not re-examine those directions already explored at the previous failed poll steps. This problem might at first seem quite straightforward, but is in fact one of the more subtle issues that must be reckoned with in the MADS framework.

One could attempt to reorient the new poll set in a pseudo-random fashion; this is in fact what was implemented in LTMADS. Though this approach will likely generate some new directions to explore with each new poll step, such an approach will also waste computational effort with some new poll points that are essentially aligned with polling directions that have already been tried (unsuccessfully) around the current CMP.

OrthoMADS thus introduced some sort of low-discrepancy 'quasi-random' Halton sequence on the first 'seed' vector used to generate the poll set, in an attempt to generate a fresh new set of polling directions. This first seed vector uniquely defines the remaining orthogonal directions of the poll set when $n = 2$. For larger n, however, it does not; by focusing only on the successive placements of the seed vector, when $n > 2$, it is not at all clear that the *entire* new poll set will be well differentiated from the previous sets of polling directions already explored around the current CMP.

In the present work, we thus propose a more geometric solution to this problem. Notably, our solution considers *all* of the directions of the failed poll sets, as well as *all* of the directions the prospective new poll set (that is, not just the seed vectors that generate these directions). The approach we use is a natural extension of the Thompson algorithm described previously. We simply add additional fixed charged particles, with substantially reduced charge, at the failed poll points from the previous (failed) poll sets when we solve the Thompson problem for the new poll points[20]. This naturally

[20] A *generalized* Thompson formulation may also be used to account for the forces applied by the fixed particles associated with the points from the previous failed poll sets, applying a force that falls off faster than the $1/r^2$ rule of normal charged particles. So doing achieves a differentiation between the old and new directions in the resulting algorithm, but tends to reduce the additional deformation of the new poll

generates a new poll set which is not only itself highly uniform, but is also generally well differentiated from the directions of the previous (failed) poll sets around the CMP, thus generalizing naturally the idea of low-discrepancy *sequences* of vectors to low-discrepancy *sets* of vectors.

Optional step: including a poll point designed to accelerate convergence when the function is locally C^1

As discussed in Chapter 3, if

- the function is locally continuously differentiable,
- the CMP is not yet at a critical point,
- there are no active constraints,
- a poll set is considered which positively spans the CMP, and
- the grid spacing is sufficiently small,

then one of the poll points is guaranteed to provide an improved function value, below that of the CMP.

If all of the above assumptions are true, except that the grid spacing is not yet quite sufficiently small enough to ensure that an improved function value is evident in the poll set (that is, if quadratic terms in the local Taylor series expansion of the cost function are still significant), then it is straightforward to estimate the linear terms of the local Taylor series expansion of the function if a poll step fails, and then to identify the downhill direction in this locally linear approximation of the function. This may be achieved simply by taking a linear fit of the function evaluations in the most recent failed poll step[21], denoted here $f(\mathbf{x}^{(i)}) = f^{(i)}$ for $i = 1,\ldots,p$ where $p \geq n+1$. Fitting these function evaluations with the linear model $f(\mathbf{x}) = \mathbf{x}\cdot\mathbf{g} + b$ and assembling the results

set that these additional fixed particles might otherwise create.

[21]The function value at the CMP itself may be ignored in this fit, because this function value does not affect the linear coefficients in local Taylor series expansion of the function.

for each of the p poll points, we may write

$$\begin{pmatrix} x_1^{(1)} & \cdots & x_n^{(1)} & 1 \\ x_1^{(2)} & \cdots & x_n^{(2)} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(p)} & \cdots & x_n^{(p)} & 1 \end{pmatrix} \begin{pmatrix} g_1 \\ \vdots \\ g_n \\ b \end{pmatrix} = \begin{pmatrix} f^{(1)} \\ f^{(2)} \\ \vdots \\ f^{(p)} \end{pmatrix}.$$

If $p = n+1$, this system of equations may be solved for the gradient $\mathbf{g}$; if $p > n+1$, a least-squares estimate of the gradient $\mathbf{g}$ is easily determined from this system of equations. Either way, the gradient so determined may be normalized and scaled by the average radius of the target shell of the subsequent poll step, and the closest lattice point on the refined grid to the negative of this vector (that is, in the downhill direction in the locally linear approximation of the function) found, thus generating what we might identify as at least a new "lattice point of interest". The subsequent poll set may thus be forced to include this new lattice point of interest (and, perhaps, scheduled to evaluate this new poll point first). Using the Thompson algorithm described previously, of course, this is quite easy to accomplish: simply add one more fixed charged particle on the sphere corresponding to this new lattice point of interest, and optimize the remaining free particles as described previously. Note that, if the function is not expected to be locally C^1 fairly often as the iteration proceeds, or if a given poll step includes one or more poll points which prove to be infeasible, then this optional step should certainly be skipped.

4.4.5 Keeping a given poll orientation if a poll successfully finds a new CMP, facilitating discrete line minimizations

A new poll set orientation is selected (and the grid refined) only after a poll step does not successfully identify a new CMP. If, on the other hand, a poll step succeeds in identifying a new CMP, then the old poll set orientation is used around the new CMP (without refining the grid), and the first direction polled is in the same direction as moved previously. Since incomplete polling is used (see §4.4.2), if this new poll point again reduces the function value, then the iteration proceeds further in this direction without evaluating the poll points in the other directions, thus allowing something of a

discrete line minimization to be performed via successive (incomplete) poll steps, all proceeding in the same direction after a single function evaluation at each poll step. This strategy, combined with the mesh coarsening heuristics discussed in §4.4.6, tends to make maximum use out of any given descent direction that may be identified, which in some problems (such as those with active constraints, or those with only piecewise differentiable functions, as discussed previously) might in fact take several failed poll steps (that is, many many function evaluations) in order to find.

4.4.6 Coarsening the mesh

Under challenging cost functions, e.g. hard constraints, Λ-MADS converges well due to the poll set orientation re-using methodology outlined above. New poll orientations are utilized until a descent direction is realized, at which point the successful orientation is maintained until the next unsuccessful Poll step. This ensures convergence when there exists a cone of descent directions, however narrow. However, as the convergence of MADS algorithms is contingent upon grid refinement after unsuccessful Poll steps, often the mesh is refined very significantly while the new Poll orientations are tried. This behavior limits the step size once an appropriate Poll orientation is located. To avoid the low convergence due to mesh overrefinement, OrthoMADS implements a mesh coarsening scheme where the mesh is coarsened by a factor of 4 at every successful Poll step. The poll orientation utilized on the coarser mesh is simply the direction previously given by the Halton sequence at that level of mesh refinement.

This behavior, while attempting to prevent mesh overrefinement, generally fails to deliver increased performance, as the Poll reverts to a nonoptimal orientation upon coarsening, virtually ensuring the failure of the Poll on the coarser mesh.

In the interest of delivering superior convergence performance, Λ-MADS implements the following coarsening scheme: upon two successive successful Poll steps utilizing the same poll orientation, Λ-MADS coarsens the mesh by a factor of 2, *maintaining the current, successful, Poll orientation*. If the next Poll proves successful, the algorithm continues as normal; if unsuccessful, the next Poll step refines the mesh by a factor of 2 *maintaining* the current Poll orientation on the finer mesh. Thus, the successful Poll orientation is located on the fine mesh and re-used on the coarse mesh. In the

case that several Poll steps are required to locate a Poll orientation lying in the cone of descent directions, this methodology allows for this orientation to be utilized on coarser meshes, taking larger steps toward the function minimum.

4.5 Isolated numerical testing of each component issue leading to Λ-MADS

In order to isolate the effects of the options presented in Section 3, several MADS algorithms, each incrementally different from the previous, were numerically tested to determine comparative convergence efficiencies. In testing the comparative performance of two algorithms, a statistically relevant number of optimizations were performed to calculate the average performance of each algorithm. In each test, the cost function consists of a randomly generated quadratic bowl. The minimum of this cost function is selected as a random point a distance of $r = 1$ from the origin. The initial CMP is a random point located a distance $r = 10$ from the origin. The lattice scaling of the Z_n lattice was set to one: $R_{Z_n} = 1$; the scaling of the Λ lattices were selected such that the volume of the voronoi cell matched the volume of the Z_n voronoi cell at the scaling above, that is, $R_{\Lambda_n} = (\Delta_{\Lambda_n}/\Delta_{Z_n})^{1/n}$.

Both optimizations begin at the initial CMP and are then converge to a tolerance of 0.001 of the initial CMP value. One thousand such runs were performed for each algorithm comparison. In comparing two algorithms A and B, the parameters quantifying performance are the percent of total runs that algorithm B converged faster than algorithm A, p, and the ratio of the average number of functional evaluations algorithm B required to the number of evaluations that algorithm A required, r. Thus, as p approaches 100 and r approaches 0, algorithm B becomes far more efficient than algorithm A.

As discussed in Section 3, in Λ-MADS there is the option to build the poll set, refining with a factor of 4, on shells $1, 2, 4, 8, 16...$ or refining by a factor of 2, on shells $1, 2, 3, 4, ...$ We thus test Λ-MADS with a minimal positive basis and fast, factor of 4, refinement, then slow, factor of two, refinement. As shown in lines 1 and 2 of Table 4.3, we find that the slow refinement scheme results in a more efficient algorithm.

Next, we investigate the effect of the lattice choice in a MADS algorithm by

Table 4.3: Convergence comparison of fundamental features of the Λ-MADS algorithm

n		2	3	4	5	6	7	8
p	$2x$ vs $4x$	61.1	56.0	54.3	52.5	56.6	57.7	57.7
r		0.828	0.905	0.872	0.921	0.874	0.868	0.896
p	Z_n vs Λ_N	45.3	50.5	54.4	54.3	56.2	64.1	66.8
r		1.11	1.01	0.946	0.948	0.930	0.873	0.8173
p	Max basis vs Min basis	47.7	53.6	65.5	78.7	85.1	91.9	94.5
r		1.36	1.37	0.868	0.602	0.481	0.369	0.276
p	Complete vs Incomplete	70.3	71.6	75.1	75.5	69.2	77.1	82.1
r		0.829	0.819	0.691	0.737	0.784	0.633	0.572
p	OrthoMADS vs Λ-MADS	44.9	51.4	56.1	78.4	74.0	79.8	84.8
r		0.839	0.865	0.821	0.638	0.81	0.607	0.48

comparing Z-MADS to Λ-MADS, both utilizing a maximal positive basis, and the slow factor-of-2 refinement discussed above. As can be seen in lines 3-4 of Table 4.3, by simply replacing the Z_n grid with the Λ_n lattice, the MADS algorithm makes significant gains in efficiency in dimensions higher than 3. In lower dimensions, as expected, the performance difference is negligible; as the dimensions increases the performance difference becomes more and more pronounced. These results indicate how efficient lattices are the preferred choice compared to the Cartesian grid for coordinating MADS optimizations, particularly as the dimension of the cost function increases.

The choice of a minimal over a maximal positive basis has not, to the authors' knowledge, been numerically established in the literature. While it has often been suggest that a minimal basis *should* increase convergence rates, we test this hypothesis in lines 5-6 of Table 4.3. The maximal basis is more efficient in low dimension ($n = 2$ and 3), as the dimension increases, the difference between the choice of basis becomes significant; the minimal basis provides superior performance to the maximal basis. For maximizing efficiency in high dimensions, a minimal positive is the appropriate configuration. As per these results, Λ-MADS is configured to utilize a maximal basis for $n < 4$

and a minimal basis for higher dimensions.

The question of incomplete polling (that is, terminating the Poll step upon locating a superior CMP) compared to complete polling (that is, evaluating the cost function on each member of the Poll set before redefining the CMP) has also remained neglected in the literature. As such, the efficiency comparison of Λ-MADS, utilizing a minimal positive basis with factor-of-2 refinement, can be seen in lines 7-8 of Table 4.3. The data clearly demonstrate how the incomplete poll set is the appropriate choice for all dimensions.

These numerical results validate the utilization in Λ-MADS of the following: building poll sets on the Delaunay cells of the Λ lattice, refining one shell per refinement (refining the mesh by a factor of 2); implementing a minimal compared to a maximal positive basis in dimensions greater than four; and using incomplete as opposed to complete polling. These features make Λ-MADS unique among MADS-type algorithms. Having tested each component leading to the definition of Λ-MADS, the numerical comparison to OrthoMADS is made. The results can be found in lines 9-10 of Table 4.3. Λ-MADS demonstrates significantly improved convergence rates compared to OrthoMADS, requiring only 48% to 90% as many function evaluations to reach convergence, and converging faster than OrthoMADS in the majority of trials, with the performance difference becoming larger as the dimension of the cost function increases.

4.6 Further numerical testing of the complete Λ-MADS algorithm

The above testing on randomly generated quadratic bowls proves valuable in evaluating the relative efficiencies of various component selections in establishing the Λ-MADS algorithm. Testing comparing to OrthoMADS indicates an increase in convergence rate. To further investigate these results, further testing was performed, precisely as described above, on the *n*-dimensional Rosenbrock cost function. The standard 2-dimensional Rosenbrock function is well known as an optimization benchmark; the deep ‘valley’ in which the optimum lies makes for a particularly challenging convex optimization problem. The analog in higher dimensions is given by

Table 4.4: Performance comparison between OrthoMADS and Λ-MADS on the Rosenbrock test function.

n	2	3	4	5	6	7	8
p	90.0	N/A	94.21	N/A	87.58	N/A	90.1
r	0.571	N/A	0.5515	N/A	0.561	N/A	0.544

$$J(x) = \sum_{i=1}^{n/2} [5(x_{2i-1}^2 - x_{2i})^2 + (x_{2i-1})^2]$$

defined only for even dimensions. This function is convex, with the global minimum at $(1,1,1,...,1)$ where the function has a value of zero.

The same series of tests described above were performed on the Rosenbrock test function in dimensions $n = 2,4,6,8$. The results can be seen in Table 4.6. This data validates the legitimacy of the previous testing on a more challenging cost function, and confirms the superior convergence rate that Λ-MADS has over OrthoMADS. As expected, the performance difference between the Cartesian-based algorithm and the Λ_n based algorithm increases with dimension. In $n = 2$, Λ-MADS requires 88% as many function evaluations to converge; in $n = 8$ it requires only 50% as many evaluations. Similarly, in $n = 2$, $r = 55$; however, in $n = 8$, $r = 97$. That is, OrthoMADS outperformed Λ-MADS in only 3% of all test optimizations in $n = 8$. This result is remarkable, and confirms the high performance of the Λ-MADS algorithm compared to its competitors.

Recall from above that the convergence metrics p and r are defined with respect to a preselected level of convergence. To test convergence rates at various levels of convergence, p and r were calculated for four differing levels of convergence: 0.1, 0.05, 0.001, 0.0001, optimizing the n-dimensional Rosenbrock function, comparing OrthoMADS to Λ-MADS. The results are graphically presented in 4.7. The superior performance of Λ-MADS indicated by the previous analysis is verified at varying levels of convergence. In $n = 4$ and greater, Λ-MADS proves to have superior convergence rates to OrthoMADS at all levels of convergence. Generally speaking, the greater the level of convergence (that is, the more difficult the optimization), the greater the performance difference between Λ-MADS and OrthoMADS.

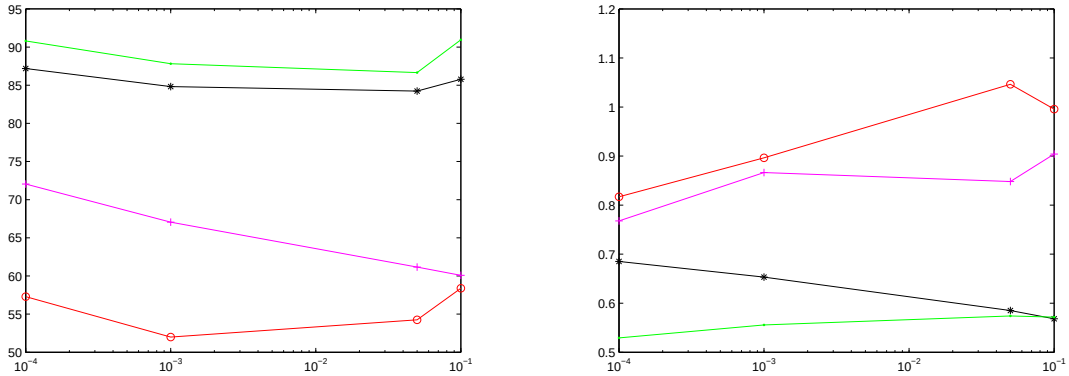

Figure 4.7: The efficiency metrics p (L) and r (R) comparing Λ-MADS to OrthoMADS on the Rosenbrock cost function for convergence goals of $0.1, 0.05, 0.001, 0.0001$ in dimensions $n = 2$ (red circles), $n = 4$ (magenta $+$), $n = 6$ (black asterisk), and $n = 8$ (green dots). Note how Λ-MADS outperforms OrthoMADS in high dimensions for all convergence levels.

Table 4.5: Performance comparison between Λ-MADS without, and with, coarsening, on quadratic bowls (lines 1-2) and the Rosenbrock test function (lines 3-4).

n	2	3	4	5	6	7	8
p	34.6	32.5	32.4	49.6	82.6	48.1	48.4
r	0.945	1.06	1.147	0.730	0.699	0.808	0.866
p	52.2	N/A	58.8	N/A	82.6	N/A	74.4
r	1.03	N/A	0.984	N/A	0.699	N/A	0.864

Finally, we test the effects of the coarsening scheme outlined above. The Λ-MADS coarsening methodology outlined above emphasizes the reuse of the successful poll orientation on the coarser grid after two consecutive successful Poll steps on the finer grid, allowing the algorithm to maintain the proper poll orientation, while taking a larger step toward the minimum, thereby maintaining a larger average step size and speeding convergence. To test the effect of coarsening in this fashion, the same testing methodology outlined above was used, testing Λ-MADS without coarsening to Λ-MADS with coarsening on a statistically relevant number of quadratic bowls, and then the n-dimensional Rosenbrock function. The results are summarized in Table 4.6.

Coarsening offers superior convergence in high dimensions, particularly on the

challenging Rosenbrock function. However, somewhat surprisingly, on these convex and unconstrained cost functions, coarsening offered no advantage in lower dimensions, in fact incurring a performance penalty. Notice that coarsening performed particularly well on the Rosenbrock function, clearly delivering superior performance than the non-coarsening algorithm. This indicates that implementing a coarsening strategy will be more valuable on a cost function with challenging behaviors. Unconstrained cost functions are comparatively easy for a MADS algorithm to handle as locating a descent direction is straightforward; more challenging is maintaining an appropriate mesh scaling in the presence of hard constraints. In the latter scenario Λ-MADS often has to perform many unsuccessful Poll steps before a descent direction can be located. While Λ-MADS' ability to refine the mesh more slowly than LTMADS and OrthoMADS prevents as much over-refinement during this process, often the mesh is refined more than necessary. Under these circumstances, the coarsening strategy is particularly appropriate. Thus, we recommend that users implement mesh coarsening on difficult constrained optimization problems.

4.7 A Numerical Example: Locating the Deep Hole of a Lattice

An example of a research optimization problem that can be solved with Λ-MADS but not by a simpler SP pattern search was encountered by the authors while performing numerical analysis of efficient lattices (see Chapter 2). The challenge is to calculate the location of a deep hole belonging to the origin node of a particular lattice. By definition, a deep hole is the furthest point from a given lattice node that remains as close or closer to said node than any other node of the lattice. Thus, if one enumerates a great number of lattice points surrounding the origin, any given point can be analyzed to determine whether or not said point lies within the voronoi cell (that is, if the point is closer to the origin than any other lattice point in the cloud). The objective is to locate the point furthest from the origin that remains in the voronoi cell of the origin node. The cost function for the A_2 lattice can be seen in Figure 3.2.

In the interest of remaining computationally feasible, the constraints must be

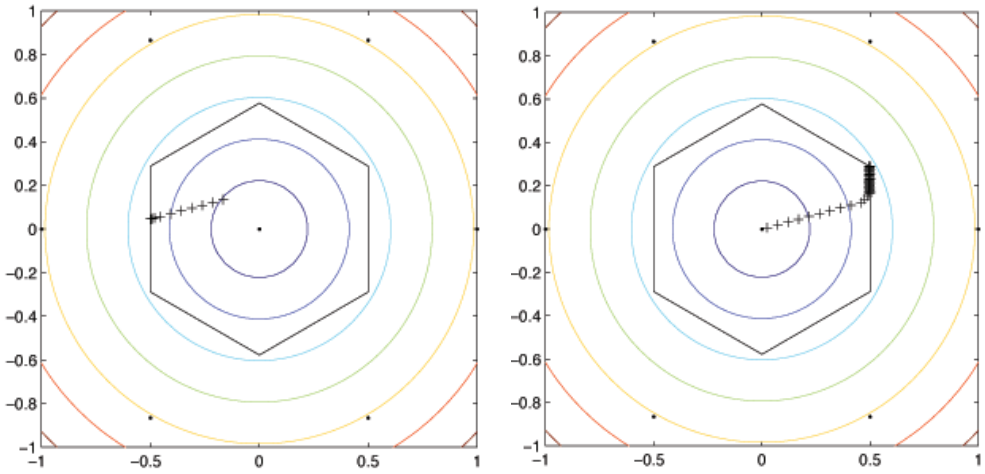

Figure 4.8: Locating the deep holes of the A_2 lattice utilizing Successive Polling (left) and Λ-MADS (right). The hard constraints are indicated in black; the cost funtion contours are plotted as well. The inability of the SP algorithm to handle constraints prevents convergence; Λ-MADS maintains convergence and locates the deep hole.

hard. This is performed by calculating the distance from each node in the cloud to the CMP. If the distance from the CMP to the origin is less than the distance from the CMP to any another node, the CMP lies inside the voronoi cell and the cost function value is the distance to the origin. Otherwise, the CMP lies outside the voronoi cell, and as such is not valid for evaluation, so the cost function value is infinity. This presents a challenging problem where traditional derivative-based algorithms cannot be applied, as the constraint surfaces are unknown, and SP and other simple GPS algorithm fail to converge.

Under the only numerically feasible problem definition, as can be seen in Figure 3.8, the Successive Polling algorithm ceases convergence upon encountering a constraint surface. Once the algorithm nears the constraint boundary, the only element of the poll set with a component in the descent direction violates the constraint and the algorithm stalls. The Λ-MADS algorithm, however, stochastically locates an orientation allowing it to follow the constraint directions and moves along the constraints to the deep hole. This method was used to locate the deep holes of a great number of lattices, allowing for the calculation of many previously unknown metrics, reported in Bewley, Belitz, &

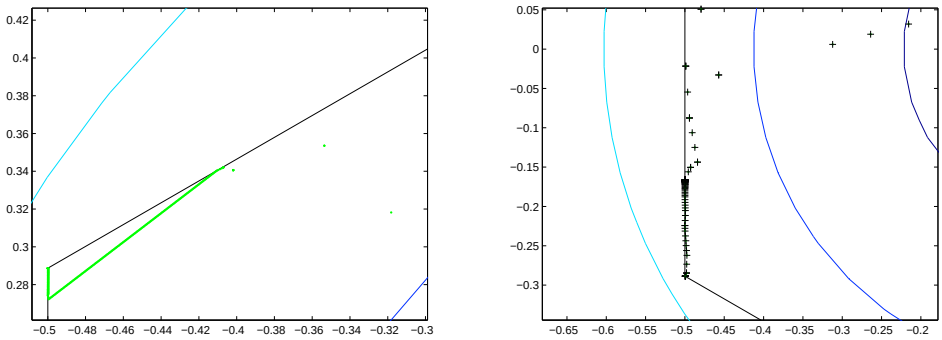

Figure 4.9: Locating the deep holes of the A_2 lattice utilizing Λ-MADS with no coarsening, plotting the CMP as green dots (L), and coarsening implemented (R), plotting the CMP as a black cross. The algorithm with coarsening enabled clearly maintains a larger average step size, speeding convergence in the presence of active constraints.

Cessna, (2011).

Figure 3.8 above clearly demonstrates one shortcoming of a non-coarsening MADS scheme on a cost function subject to hard constraints: while the algorithm locates a suitable descent direction, the mesh becomes very fine, limiting the step size taken. To rectify this, the coarsening scheme described above is implemented in Λ-MADS, and the deep hole test function is reconsidered. As can be seen in Figure 3.8, without coarsening, the step size along the black constraint boundary is very small, requiring a great number of function evaluations to converge. With coarsening, good descent directions are reused, and the deep hole is located while maintaining a coarser grid size on average.

4.8 Conclusion

In this document we investigate the performance of current Mesh Adaptive Direct Search (MADS) methods, and introduce a new MADS algorithm, Λ-MADS. Via exhaustive numerical testing, we conclusively demonstrate that in the interest of algorithm efficiency, it is highly desirable to coordiate a MADS search on (1) an efficient lattice, (2) to locate the Poll sets on the Delauney cell of the lattice, (3) to utilize a min-

imal rather than a maximal positive basis, and (4) to implement incomplete polling of the Poll sets.

The Λ-MADS algorithm is significantly more efficient than any competing MADS algorithm due to careful selection of the most efficient subset of the choices presented above. As such, Λ-MADS is the most efficient MADS algorithm yet developed, and is the clear choice for difficult modern convex optimization problems.

Chapter 5

Conclusion

The current work investigates the application of n-dimensional sphere packings, or lattices, as they relate to modern derivative-free optimization algorithms. The results presented in Chapters 3 and 4 demonstrate clearly that significant gains in convergence rates are commonly realized among the best modern grid-based algorithms by replacing the ubiquitous Cartesian grid with an efficient lattice.

In Chapter 2 a review of sphere packing theory is presented in a cogent fashion, covering the majority of well known lattices, giving thorough characterizations of said lattices, including significant new material. Several new packings are introduced, some specifically for engineering applications, and many previously unknown metrics are presented, rounding out the analysis of common lattices. Several higher-dimensional extensions of previously known packings are also considered. Numerous algorithms necessary for practical implementation of lattice theory are presented, most importantly quantization algorithms for the most efficient members of the Λ class of lattices. Finally, a summary of coding theory, the most widely known application of n-dimensional sphere packings to engineering applications, is presented. Chapter 2 aims to be a concise, engineering-oriented presentation of lattice theory, aimed to motivate and guide future work applying this rigorous field of mathematics to the modern fields of engineering and computational science.

Chapter 3 delves into modern derivative-free optimization theory and implementation, investigating the Generalized Pattern Search (GPS) class of algorithms. The Surrogate Management Framework (SMF) algorithm is introduced and investigated, start-

ing with the Successive Polling (SP) pattern search Poll step. Adapting the SP algorithm to the efficient lattices presented in Chapter 2 is covered, and extensive numerical testing indicates that simply by replacing the inefficient Cartesian grid with a more efficient alternative the SP algorithm's convergence rate is improved due to greater radial and angular uniformity of the poll set. Next, the Search step of the SMF algorithm is explored, reviewing Kriging surrogate methodology and derivation. With an understanding of Kriging, various globally and nonlocally convergent Search schemes are described and tested. One of the most effective modern Search algorithm maximizes the likelyhood of improvement, giving a Search step that is both globally convergent and locally efficient. Combining the highly efficient lattice-based SP Poll step and the MLI Kriging Search step, a new SMF algorithm, named LAttice-Based Derivative-free Optimization via Global Surrogates (LABDOGS) algorithm is defined. Numerical testing of LABDOGS indicates that, as expected, the lattice-based SMF code significantly outperforms its Cartesian equivalent, with the performance difference becoming greater as the dimension of the problem increases. The MLI search offers global convergence as well as speeding local convergence, and the efficient lattice-based SP Poll step ensures fast local convergence.

Motivated by the success of implementing efficient lattices in the SMF framework, Chapter 4 focuses on a more sophisticated class of GPS algorithms known as Mesh Adaptive Direct Search (MADS) algorithms. Among the most sophisticated of modern GPS algorithms, MADS algorithms aim to improve upon the convergence behavior exhibited by simpler GPS methods. Specifically, constrained optimization problems present a special challenge to SP algorithms, which fail to converge when a hard constraint prevents any valid poll directions from lying in the cone of descent directions. MADS, specifically OrthoMADS, overcomes this problem by allowing the polling direction to become dense as convergence is neared. Combining the dense polling directions with intelligent poll orientation re-using, a well designed MADS algorithm converges well even in the presence of constraints or other challenging function behavior that foils conventional GPS algorithms. Recognizing that the best of modern MADS algorithms, OrthoMADS, is intrinsically limited by several factors, including its design upon the Cartesian lattice, Chapter 4 builds a new MADS algorithm, Λ-MADS,

from the ground up, numerically and theoretically investigating every option available to build the algorithm. In comparison to previous MADS algorithms, Λ-MADS achieves improved convergence performance by implementing an efficient lattice, building more uniform poll sets, refining the poll set via new methodology, utilizing a minimal basis when appropriate, and finally implementing a more effective coarsening strategy than had previously been described in the literature. All these changes, numerically isolated, tested, and verified, result in Λ-MADS inheriting all the convergence behavior present in the best MADS algorithms, while increasing the convergence rate significantly. In $n = 8$, for example, Λ-MADS requires approximately half as many function evaluations as OrthoMADS to reach the same level of convergence. As with LABDOGS, as the dimension of the cost function increases, the performance difference between Λ-MADS and OrthoMADS increases, establishing Λ-MADS as the most efficient MADS algorithm yet devised.

The application of n-dimensional sphere packings to modern optimization algorithm design is only one example of the impacts that this field of mathematics can potentially have a significant impact in. Coding theory, as summarized in Chapter 2, is another example; using lattices to provide an efficient interconnect scheme for supercomputers as described in Cessna & Bewley (2011) is another. However, the potential for engineering application of lattice theory is vast. The author hopes that the material presented in Chapter 2 will prove useful in achieving this potential, with the applications decribed in Chapters 3 and 4 serving as motivation.

The most logical extension of the work presented in this document is the merging of the LABDOGS and Λ-MADS algorithms; by replacing the SP Poll step of LABDOGS with a MADS step, a hybrid algorithm, perhaps to be named MADDOGS, will maintain all the convergence characteristics of LABDOGS while offering superior constraint handling abilities. As LABDOGS by nature requires a bounded domain for the MLI search to be defined, choosing a Poll step that can best handle such constraints would be a efficacious strategy.

Further refinements of LABDOGS or the aforementioned MADDOGS will probably be minor. One weakness of LABDOGS is its use of Ordinary Kriging. The assumptions behind the Kriging model have the surrogate converge to the mean of the

data points where the surrogate is far from previously explored regions. Additionally, Ordinary Kriging fails to account for variability in individual dimensions, limiting the ability of the surrogate to accurately model difficult functions. Perhaps some convergence potential in LABDOGS could be realized by implementing more sophisticated forms of Kriging, or perhaps a different interpolating strategy altogether. In practice, it is difficult to imagine how a Poll step more effective than Λ-MADS would be generated beyond the modifications outlined in Chapter 3 (e.g. using the surrogate to predict the best Poll point, which could be extended to using the surrogate to predict an appropriate Poll orientation). Similarly, while the MLI Search is not the most sophisticated Search available, real-world performance between MLI and other Search algorithms is likely to be difficult to measure.

The Λ-MADS algorithm is quite complete and mature for the cost functions herein assumed, laying some fundamental questions relevant to GPS algorithms in general to rest (specifically, the effect of utilizing a maximal versus minimal positive basis). While the MADS class of algorithms was specifically developed for challenging function behaviors, e.g. hard constraints, little work has been done in establishing discrete analogs to modern derivitive-based optimization constraint handling, which typically optimize on a subspace of constraints, adding and removing active constraints as the algorithm progresses toward an optimum. This sort of behavior significantly reduces the dimension of the space being considered by the optimization algorithm. As such, an obvious extension in derivative-free algorithms would allow the Poll set to be redefined on a reduced-dimension subspace of constraint ‘walls’ when appropriate. This idea is outlined in Chapter 3 for a SP optimization, the same could be implemented in MADS. Defining the poll set on the subspace of active constraints, and adding appropriate vectors allowing the algorithm to move off the constraints could possibly offer significant improvements in convergence rates on cost functions with linear constraints.

Further extension of this work, in the mind of the author, would move away from the prototypical cost function assumed throughout this work, the distinguishing characteristics of which include continuity and coarse-scale differentiability. One fundamental limitation of the GPS class of algorithms herein presented is that for discrete or integer cost functions (e.g. the famous Travelling Salesman problem), GPS methods

are not generally applicable. Simulated Annealing or Genetic algorithms, by comparison, can handle such cost functions; indeed, while SA and GA optimizations are often (mis) applied to continuous cost functions, they are best suited to discrete problems. The question of how to extend the lattice-based methodology presented in this document to such new classes of optimization problems is open for research, and may well result in interesting and significant findings.

In the meantime, it is the author's sincere hope that LABDOGS, Λ-MADS, and the presentation of *n*-dimensional sphere packings of Chapter 2 and Bewley, Belitz, & Cessna (2011), will find widespread application in modern engineering.

Bibliography

[1] Abramson, MA, Audet, C, Dennis, Jr, JE, & S. Le Digabel (2009) OrthoMads: A deterministic Mads instance with orthogonal directions. *SIAM Journal on Optimization*, **20**, 948-966.

[2] Agrell, E, & Eriksson, T (1998) Optimization of Lattices for Quantization. *IEEE Transactions on Information Theory* **44**, 1814-1828.

[3] Anzin (2002) On the density of a lattice covering for n = 11 and n = 14 (in Russian), *Uspekhi Mat. Nauk* **57**, no. 2, 187âĂŞ188; English translation in *Math. Surveys* **57**, 407âĂŞ409.

[4] Aste, T, & Weaire, D (2008) *The pursuit of the perfect packing.* Taylor & Francis.

[5] Audet, C, & Dennis, Jr, JE (2006) Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, **17** 188âĂŞ217.

[6] Baranovskii, EP (1994) The perfect lattices $\Gamma(A^n)$, and the covering density of $\Gamma(A^n)$. *European Journal of Combinatorics* **15**, 317-323.

[7] Belitz, P, & Bewley, T (2011) New Horizons in Sphere Packing Theory, Chapter 1I: Lattice Based Derivative Free Optimization via Global Surrogates. *SIAM Journal on Optimization*, submitted.

[8] Bewley, T, Belitz, P, & Cessna, J (2011) New Horizons in Sphere Packing Theory, Chapter 1: Fundamental Concepts & Constructions, from Dense to Rare. *SIAM Review*, submitted. `http://fccr.ucsd.edu/pubs/BBC_Part1.pdf`

[9] Berrou, C, Glavieux, A, & Thitimajshima, P (1993) Near Shannon limit error-correcting coding and decoding: Turbo-Codes. In *ICC'93*, Geneva, Switzerland. 1064-1070.

[10] Beukemann, A, & Klee, WE (1992) Minimal nets. *Z. Krist.* **201**, 37-51.

[11] Blatov, VA (2006) Multipurpose crystallochemical analysis with the program package TOPOS. *IUCr CompComm Newsletter* **7**, 4âĂŞ38.

[12] Blatov, VA (2007) Topological relations between three-dimensional periodic nets. I. Uninodal nets. *Acta Cryst.* **A63**, 329-343.

[13] Blatov, VA, Delgado-Friedrichs, O, O'Keeffe, M, & Proserpio, DM (2007) Three-periodic nets and tilings: natural tilings for nets. *Acta Cryst.* **A63**, 418-425.

[14] Blatov, VA, O'Keeffe, M, & Proserpio, DM (2009) Vertex-, face-, point-, Schläfli-, and Delaney-symbols in nets, polyhedra and tilings: recommended terminology. *CrystEngComm.*, 2010, DOI: 10.1039/b910671e.

[15] Blichfeldt, HF (1935) The minimum values of positive quadratic forms in six, seven and eight variables. *Math. Z.* **39**, 1âĂŞ15.

[16] Bonneau, C, Delgado-Friedrichs, O, O'Keeffe, M, & Yaghi, OM (2004) Three-periodic nets and tilings: minimal nets. *Acta Cryst.* **A60**, 517-520.

[17] Booker, A, Dennis, JR, Frank, P, Serafini, D, Torczon, V, & Trosset, M (1999) A rigorous framework for optimization of expensive functions by surrogates. *Structural and Multidisciplinary Optimization* **17**, 1âĂŞ13.

[18] Brown, H, Bülow, R, Neubüser, J, Wondratschek, H, & Zassenhaus, H (1978) *Crystallographic groups of 4-dimensional space.* Wiley.

[19] Cecil, T (2005) A numerical method for computing minimal surfaces in arbitrary dimension. *J. Comp. Phys.* **206**, 650-660.

[20] Cessna, J, & Bewley, T (2011) New Horizons in Sphere Packing Theory, Chapter 1II: Structured Computational Interconnects. *SIAM Journal on Computing*, submitted.

[21] Clark, WE (1930) *The Aryabhatiya of Aryabhata: An Ancient Indian Work on Mathematics and Astronomy*. University of Chicago Press.

[22] Cohn, H, & Kumar, A (2009) Optimality and uniqueness of the Leech lattice among lattices, *Annals of Mathematics* **170**, 1003-1050.

[23] Conway, JH, & Sloane, NJA (1984) On the Voronoï regions of certain lattices. *SIAM J. Alg. Disc. Meth.* **5**, 294-302.

[24] Conway, JH, & Sloane, NJA (1988) Low-dimensional lattices. I Quadratic forms of small determinant. *Proc. R. Soc. Lond. A* **418**, 17-41.

[25] Conway, JH, & Sloane, NJA (1997) Low-dimensional lattices. VII Coordination sequences. *Proc. R. Soc. Lond. A* **453**, 2369-2389.

[26] Conway, JH, & Sloane, NJA (1999) *Sphere Packings, Lattices, and Groups*, Springer.

[27] Coope, ID, & Price, CJ (2001) On the convergence of grid-based methods for unconstrained optimization. *SIAM J. Optim.*, **11**, 859âĂŞ869.

[28] Cox, DD & John, S (1997) SDO: A statistical method for global optimization. In *Multidisciplinary Design Optimization: State of the Art* (edited by Alexandrov, N, & Hussaini, MY), 315âĂŞ329. SIAM.

[29] Coxeter, HSM (1951) Extreme forms. *Canadian Journal of Mathematics*, **3**, 391-441.

[30] Coxeter, HSM (1970) *Twisted Honeycombs*. Regional Conference Series in Mathematics, No. 4, American Mathematical Society, Providence.

[31] Coxeter, HSM (1973) *Regular Polytopes*. Dover.

[32] Coxeter, HSM (1974) *Regular Complex Polytopes*. Cambridge.

[33] Coxeter, HSM (1987) *Projective Geometry*. Springer.

[34] Coxeter, HSM (1989) *Introduction to Geometry*. Wiley.

[35] Croft, Falconer, & Guy (1991) *Unsolved Problems in Geometry.* Springer.

[36] Curtis, RT (1976) A new combinatorial approach to M24. *Math. Proc. Camb. Phil. Soc.* **79**, 25-42.

[37] Delgado-Friedrichs, O, Foster, MD, O'Keeffe, M, Proserpio, DM, Treacy, MMJ, & Yaghi, OM (2005) What do we know about three-periodic nets? *Journal of Solid State Chemistry* **178**, 2533-2554.

[38] Delgado-Friedrichs, O, & Huson, DH (2000) 4-Regular Vertex-Transitive Tilings of E^3. *Discrete & Computational Geometry* **24**, 279-292.

[39] Delgado-Friedrichs, O, & O'Keeffe, M (2003) Identification of and symmetry computation for crystal nets. *Acta Cryst.* **A59**, 351-360.

[40] Delgado-Friedrichs, O, & O'Keeffe, M (2007) Three-periodic nets and tilings: face-transitive tilings and edge-transitive nets revisited. *Acta Cryst.* **A63**, 344-347.

[41] Delgado-Friedrichs, O, O'Keeffe, M, & Yaghi, OM (2003a) Three-periodic nets and tilings: regular and quasiregular nets. *Acta Cryst.* **A59**, 22-27.

[42] Delgado-Friedrichs, O, O'Keeffe, M, & Yaghi, OM (2003b) Three-periodic nets and tilings: semiregular nets. *Acta Cryst.* **A59**, 515-525.

[43] Delgado-Friedrichs, O, O'Keeffe, M, & Yaghi, OM (2003c) The $CdSO_4$, rutile, cooperite, and quartz dual nets: interpenetration and catenation. *Solid State Sciences.* **5**, 73-78.

[44] Delgado-Friedrichs, O, O'Keeffe, M, & Yaghi, OM (2006) Three-periodic nets and tilings: edge-transitive binodal structures. *Acta Cryst.* **A62**, 350-355.

[45] Dorozinski, TE, & Fischer, W (2006) A novel series of sphere packings with arbitrarily low density. *Z. Kristallogr.* **221**, 563-566.

[46] Elder, JF IV (1992) Global Rd optimization when probes are expensive: the GROPE algorithm. *Proceedings of the 1992 IEEE International Conference on Systems, Man, and Cybernetics,* **1**, 577âĂŞ582, Chicago.

[47] Fejes Tóth, L (1971) Perfect distribution of points on a sphere. *Periodica Mathematica Hungarica* **1**, 25-33.

[48] Fejes Tóth, L (1959) Verdeckung einer Kugel durch Kugeln. *Publ. Math. Debrecen* **6**, 234-240.

[49] Fejes Tóth, L (1972) *Lagerungen in der Ebene auf der Kugel und im Raum.* Springer.

[50] Fejes Tóth, L (1975) Research problem 13. *Period. Math. Hungar.* **6**, 197-199.

[51] Fincke, U, & Pohst, M (1985) Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Math. Comp.* **44**, 463-471.

[52] Fischer, W (2005) On sphere packings of arbitrarily low density. *Z. Kristallogr.* **220**, 657âĂŞ662.

[53] Friedman, E (2009) Erich's Packing Center: http://www2.stetson.edu/∼efriedma/packing.html.

[54] Gallager, RG (1963) *Low Density Parity Check Codes*. Monograph, M.I.T. Press.

[55] Gandini, PM, & Willis, JM (1992) On finite sphere packings. *Math Pannon* **3**, 19-29.

[56] Gandini, PM, & Zucco, A (1992) On the sausage catastrophe in 4-space. *Mathematicka* **39**, 274-278.

[57] Gauss, CF (1831) Untersuchungen über die Eigenscahften der positiven ternären quadratischen Formen von Ludwig August Seber, *Göttingische gelehrte Anzeigen*, 1831 Juli 9, also published in *J. Reine Angew. Math.* **20** (1840), 312âĂŞ320, and *Werke*, vol. 2, Königliche Gesellschaft der Wissenschaften, Göttingen, 1876, pp. 188âĂŞ196.

[58] Gensane, T (2004) Dense packings of equal spheres in a cube. *Electronic J. Combinatorics* **11** (1), #R33, 1-17.

[59] Grosse-Kunstleve, RW, Brunner, GO, & Sloane, NJA (1996) Algebraic description of coordination sequences and exact topological densities for zeolites. *Acta Cryst.* A52, 879-889.

[60] Grover, P, Sahai, A, & Park, SY (2010) The finite-dimensional Witsenhausen counterexample. *IEEE Transactions on Automatic Control*, submitted. Draft available as: arXiv:1003.0514v1.

[61] Hales, S (1727) *Vegetable Staticks*, London: Printed for W. and J. Innys; and T. Woodward.

[62] Hales, TC (2005) A proof of the Kepler conjecture. *Annals of Mathematics,* **162**, 1065âĂŞ1185.

[63] Hales, TC (2006), Historical overview of the Kepler conjecture, *Discrete Comput Geom* **36**, 5âĂŞ20.

[64] Harriot, T (1614) *De Numeris Triangularibus Et inde De Progressionibus Artithmeticis: Magisteria magna*, in Berry, J, and Stedall, J, editors (2009) *Thomas Harriot's Doctrine of Triangular Numbers: the 'Magisteria Magna'*, published by the European Mathematical Society.

[65] Heath, TL (1931) *A Manual of Greek Mathematics*. Oxford University Press (republished in 2003 by Dover).

[66] Heesch, H, & Laves, RT (1933) Über dünne Kugelpackeungen, *Z. Krist.* **85**, 443-453.

[67] Hyde, ST, Delgado-Friedrichs, O, Ramsden, SJ, Robins, V (2006) Towards enumeration of crystalline frameworks: The 2D hyperbolic approach. *Solid State Sci.* **8**, 740âĂŞ752.

[68] Isaaks, EH, & Srivastav, RM (1989) *An Introduction to Applied Geostatistics.* Oxford.

[69] Janssen, T (1986) Crystallography of quasi-crystals. *Acta Cryst.* **A42**, 261-271.

[70] Jones, DR (2001) A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization* **21**, 345-383.

[71] Kaku, M (1999) *Introduction to Superstring and M-Theory*. Springer-Verlag.

[72] Kelvin, WT (1887) On the Division of Space with Minimum Partitional Area, *Philosophical Magazine* **24**, No. 151, p. 503.

[73] Kepler, J (1611) *Strena seu de nive sexangula* (*The six-cornered snowflake*).

[74] Koch, E, & Fischer, W (1995) Sphere packings with three contacts per sphere and the problem of the least dense sphere packing. *Z. Krist.* **210**, 407âĂŞ414.

[75] Koch, E, & Fischer, W (2006) Sphere packings and packings of ellipsoids. In *International Tables for Crystallography*, Vol C, Sec 9.1.1, pp 746-751.

[76] Korkine, A, & Zolotareff, G (1873) Sur les formes quadratiques. *Math. Ann.* **6**, 366âĂŞ389.

[77] Korkine, A, & Zolotareff, G (1875) Sur les formes quadratiques positives, *Math. Ann.* **11**, 242âĂŞ292.

[78] Krige, DG (1951) *A statistical approach to some mine valuations and allied problems at the Witwatersrand*. Master's thesis of the University of Witwatersrand, South Africa.

[79] Kushner, HJ (1964) A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, **86**, 97âĂŞ106.

[80] Lagrange, JL (1773) Recherches dâĂŹarithmétique. *Nov. Mem. Acad. Roy. Sc. Bell Lettres Berlin* In *Œuvres*, **3**, Gauthier-Villars, Paris, 1867âĂŞ1892, pp. 693âĂŞ758.

[81] Leech, J (1956) The problem of the thirteen spheres. *Math. Gazette* **40**, 22-23.

[82] Leech, J, & Sloane, NJA (1971) Sphere Packing and Error-Correcting Codes. *Canad. J. Math.* **23**, 718-745.

[83] Matheron, G (1963) Principles of geostatistics. *Economic Geology* **58**, 1246-1266.

[84] Meschkowski, H (1960) *Ungel"oste und unl"osbare Probleme der Geometrie*, Braunsehweig.

[85] Milnor, J (1976) HilbertâĂŹs problem 18: on crystallographic groups, fundamental domains, and on sphere packings, in *Mathematical Developments Arising from Hilbert Problems*, Proceedings of Symposia in Pure Mathematics, vol. 28, AMS, Providence, RI, pp. 491âĂŞ506.

[86] Mockus, J (1994) Application of Bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, **4**, 347âĂŞ365.

[87] Moon, T (2005) *Error Correction Coding, Mathematical Methods and Algorithms*. Wiley.

[88] Ockwig, NW, Delgado-Friedrichs, O, O'Keeffe, M, & Yaghi, OM (2005) Recticular Chemistry: Occurance and Taxonomy of Nets and Grammar for the Design of Frameworks. *Acc. Chem. Res.* **38**, 176-182.

[89] O'Keeffe, M (1991a) Dense and rare four-connected nets. *Z. Krist.* **196**, 21-37

[90] O'Keeffe, M (1991b) *N*-dimensional diamond, sodalite, and rare sphere packings. *Acta Cryst.* **A47**, 748-753.

[91] O'Keeffe, M (2008) Three-periodic nets and tilings: regular and related infinite polyhedra. *Acta Cryst.* **A64**, 425-429.

[92] O'Keeffe, M, Eddaoudi, M, Li, H, Reineke, T, & Yaghi, OM (2000) Frameworks for Extended Solids: Geometrical Design Principles. *Journal of Solid State Chemistry* **152**, 3-20.

[93] O'Keeffe, M, Peskov, MA, Ramsden, SJ, & Yaghi, OM (2008) The Recticular Chemistry Structure Resource (RCSR) Database of, and Symbols for, Crystal Nets. *Acc. Chem. Res.* **41**, 1782-1789.

[94] Nelder, JA, & Mead, R (1965) A simplex method for function minimization *Computer Journal* **7**, 308âĂŞ313.

[95] Perttunen, C (1991) A computational geometric approach to feasible region division in constrained global optimization. *Proceedings of the 1991 IEEE Conference on Systems, Man, and Cybernetics.*

[96] Pless, V (1998) *Introduction to the Theory of Error-Correcting Codes*. Wiley.

[97] Rasmussen, CE, & Williams, CKI (2006) *Gaussian processes for machine learning.* MIT Press.

[98] Rukeyser, M (1972) *The Traces of Thomas Hariot.* Random House.

[99] Sadoc, JF, & Rivier, N, editors (1999) *Foams and Emulsions.* Kluwer.

[100] Schrijver, A (1986) *Thoery of linear and integer programming.* Wiley.

[101] Schürmann, A (2006) On packing spheres into containers; about Kepler's finite sphere packing problem. *Documenta Mathematica* **11**, 393âĂŞ406.

[102] Schürmann, A, & Vallentin, F (2006) Computational approaches to lattice packing and covering problems, *Discrete Comput. Geom.* **35**, 73-116.

[103] Schütte, K, & van der Waerden, BL (1953) Das Problem der dreizehn Kugeln. *Math Annalen* **125**, 325-334.

[104] Schwarzenberger, RLE (1980) *N-dimensional crystallography.* Pitman.

[105] Sikirić, MD, Schürmann, A, & Vallentin, F (2008) A generalization of Voronoi's reduction theory and its application. *Duke Mathematical Journal* **142**, 127-164.

[106] Skelton, RE, & de Oliveira, MC (2009) *Tensegrity Systems.* Springer.

[107] Sloan, IH, & Kachoyan PJ (1987) Lattice methods for multiple integration: theory, error analysis and examples. *SIAM J. Num. Anal.* **24**, 116-128.

[108] Sloane, NJA (1987) Theta-Series and Magic Numbers for Diamond and Certain Ionic Crystal Structures, *J. Math. Phys.* **28**, 1653-1657.

[109] Spendley, W, Hext, GR, & Himsworth, FR (1962) Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation *Technometrics* **4**, 441-461.

[110] Stuckman, BE (1988) A global search method for optimizing nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics*, **18**, 965âĂŞ977.

[111] Szpiro, G (2003) *Kepler's conjecture: how some of the greatest minds in history helped solve one of the oldest math problems in the world*. Wiley.

[112] Tammes, PML (1930) On the origin of number and arrangement of the places of exit on the surface of pollen-grains. *Recueil des travaux botaniques néerlandais* **27**, 1-84.

[113] Thomson, JJ (1904) On the Structure of the Atom: an Investigation of the Stability and Periods of Oscillation of a number of Corpuscles arranged at equal intervals around the Circumference of a Circle; with Application of the Results to the Theory of Atomic Structure. *Philosophical Magazine Series 6*, **7** (39), 237–265.

[114] Thompson, TM (1983) *From error-correcting codes through sphere packings to simple groups.* The Mathematical Association of America.

[115] Thue, A (1892) Om nogle geometrisk taltheoretiske Theoremer, *Forand. Skand. Natur.* **14**, 352âĂŞ353.

[116] Tietäväinen, A (1973) On the nonexistence of perfect codes over finite fields, *SIAM J. Appl. Math*. **24**, 88-96.

[117] Torczon, V (1997) On the convergence of pattern search algorithms. *SIAM J. Optim.*, **7**, 1-25.

[118] Torn, A, & Zilinskas, A (1987) *Global Optimization*, Springer.

[119] Treacy, MMJ, Rivin, I, Balkovsky, E, Randall, KH, Foster, MD (2004) Enumeration of periodic tetrahedral frameworks. II. Polynodal graphs. *Microporous Mesoporous Mater*. **74**, 121âĂŞ132.

[120] Weaire D & Phelan R (1994) A counterexample to Kelvin's conjecture on minimal surfaces, *Phil. Mag. Lett.* **69**, 107âĂŞ110.

[121] Wells, AF (1977) *Three-Dimensional Nets and Polyhedra*. Wiley.

[122] Wells, AF (1979) *Further Studies of Three-dimensional Nets*. ACA Monograph No. 8.

[123] Wells, AF (1983) Six New Three-Dimensional 3-Connected Nets $4.n^2$. *Acta Cryst.* **B39**, 652-654.

[124] Wells, AF (1984) *Structural inorganic chemistry*. Oxford University Press.

[125] Willis, JM (1983) Research problem 35. *Period. Math. Hungar.* **14**, 312-314.

[126] Zong, C (1999) *Sphere Packings*. Springer.