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

Paul Belitz and Thomas Bewley

Abstract

This paper 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 be drawn from a 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.

I. Background

Practical applications in engineering, science, finance, business, and elsewhere often call for efficient derivative-free algorithms for the optimization of expensive nonsmooth 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 heavily 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 Bewley, Belitz, & Cessna (2011) 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 Bewley, Belitz, & Cessna 2011) 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 $\tau = 48$, whereas
- the Leech lattice, Λ_{24}, is characterized by $\Delta = 0.001930$, $\Theta = 7.904$, $G = 0.06577$, and $\tau = 196560$.

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 Bewley, Belitz, & Cessna (2011); some of the salient properties of these lattices are compared with the corresponding Cartesian grids in Table 1, the $\mathbb{Z}^2$ and Λ_2 lattices are visualized in Figure 1, and the $\mathbb{Z}^3$ and Λ_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.

lattice	Δ	Θ	G	τ	Available points to select the poll set from as the grid is refined.
$D_2 \cong \mathbb{Z}^2$	0.78540	1.5708	0.08333	4	L/O: 8, 16, 32, 64, 128, … (see Figures 1a, 3, and 4) $\mathbb{Z}$: 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, … (see Figures 1b and 5)
$A_2 \cong \Lambda_2$	0.90690	1.2092	0.08019	6	Λ: 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, … (see Figures 1c and 6)
$\mathbb{Z}^3$	0.52360	2.7207	0.08333	6	L/O: 26, 98, 386, 1538, 6146, … (see Figure 2a) $\mathbb{Z}$: 6, 18, 38, 66, 102, 146, 198, 258, 326, 402, … (see Figure 2b)
$D_3 \cong A_3 \cong \Lambda_3$	0.74048	2.0944	0.07874	12	Λ: 12, 42, 92, 162, 252, 362, 492, 642, 812, 1002, … (see Figure 2c)
$\mathbb{Z}^4$	0.30843	4.9348	0.08333	8	L/O: 80, 544, 4160, 32896, 262400, … $\mathbb{Z}$: 8, 32, 88, 192, 360, 608, 952, 1408, 1992, 2720, …
$D_4 \cong \Lambda_4$	0.61685	2.4674	0.07660	24	Λ: 24, 144, 456, 1056, 2040, 3504, 5544, 8256, 11736, 16080, …
$\mathbb{Z}^5$	0.16449	9.1955	0.08333	10	L/O: 242, 2882, 42242, 660482, 10506242, … $\mathbb{Z}$: 10, 50, 170, 450, 1002, 1970, 3530, 5890, 9290, 14002, …
$D_5 \cong \Lambda_5$	0.46526	4.5977	0.07579	40	Λ: 40, 370, 1640, 4930, 11752, 24050, 44200, 75010, 119720, 182002, …
$\mathbb{Z}^6$	0.08075	17.441	0.08333	12	L/O: 728, 14896, 413792, 12746944, 403964288, … $\mathbb{Z}$: 12, 72, 292, 912, 2364, 5336, 10836, 20256, 35436, 58728, …
$E_6 \cong \Lambda_6$	0.37295	7.0722	0.07435	72	Λ: 72, 1062, 6696, 26316, 77688, 189810, 405720, 785304, 1408104, 2376126, …
$\mathbb{Z}^7$	0.03691	33.498	0.08333	14	L/O: 2186, 75938, 3959426, 239479298, 15105828866, … $\mathbb{Z}$: 14, 98, 462, 1666, 4942, 12642, 28814, 59906, 115598, 209762, …
$E_7 \cong \Lambda_7$	0.29530	13.810	0.07323	126	Λ: 126, 2898, 25886, 133506, 490014, 1433810, 3573054, 7902594, 15942206, 29896146, …
$\mathbb{Z}^8$	0.01585	64.939	0.08333	16	L/O: 6560, 384064, 37281920, 4412866816, 553517580800, … $\mathbb{Z}$: 16, 128, 688, 2816, 9424, 27008, 68464, 157184, 332688, 658048, …
$E_8 \cong \Lambda_8$	0.25367	4.0587	0.07168	240	Λ: 240, 9120, 121680, 864960, 4113840, 14905440, 44480400, 114879360, 265422960, 561403680, …

TABLE I: Characteristics up to $n = 8$ of the Cartesian grid $\mathbb{Z}^n$ as compared with the Λ_n lattice. Listed first are the packing density Δ, covering thickness Θ, mean squared quantization error per dimension G, and kissing number τ, all of which indicate the lattice uniformity; note that Λ_n outperforms $\mathbb{Z}^n$ in every metric, with the differences becoming especially pronounced as n is increased. The last column indicates the number of available points to select the poll set from at the k'th level of grid refiniment; 'L/O:' denotes the LTMADS or OrthoMADS contexts (§I-C), '$\mathbb{Z}$:' denotes the $\mathbb{Z}$-MADS context (§I-D), and 'Λ:'denotes the Λ-MADS context (§I-E and §III).

lattices are visualized in Figure 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].

A. 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

[3]Note that 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 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.

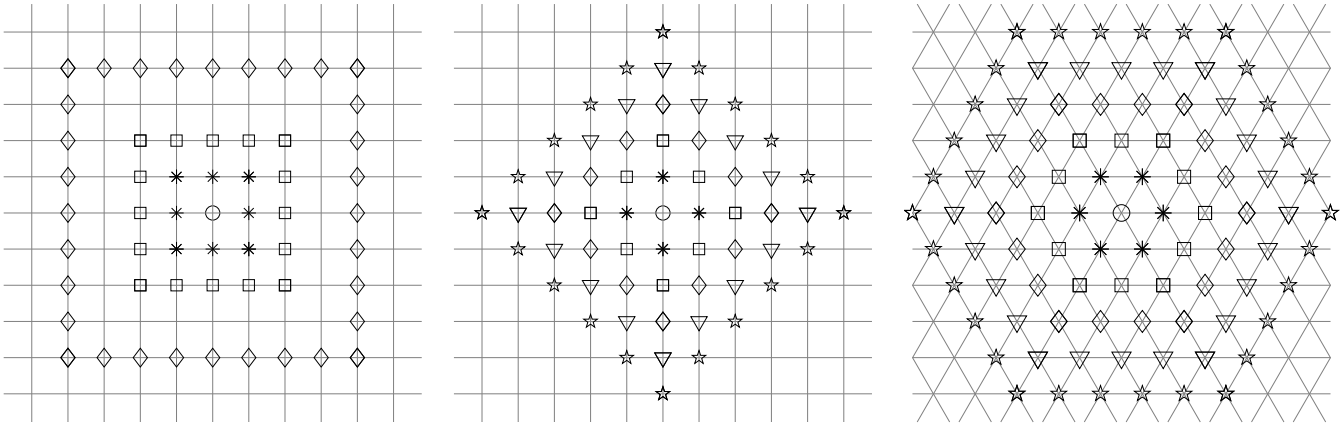

Fig. 1: The (a, b) $\mathbb{Z}^2$ and (c) Λ_2 lattices, indicating (a) the first three shells of potential poll points on $\mathbb{Z}^2$ used in the LTMADS and OrthoMADS formulations, and (b, c) the first five shells of potential poll points used, respectively, in the $\mathbb{Z}$-MADS and Λ-MADS formulations. The number of points in all three sets of shells (arranged, respectively, in squares, diamonds, and hexagons around the CMP) are listed in Table 1.

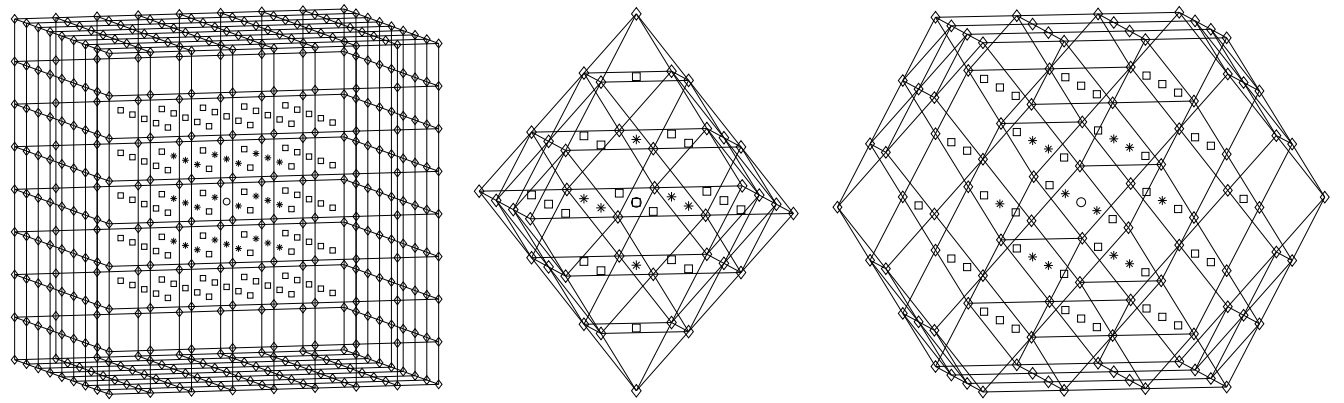

Fig. 2: The (a, b) $\mathbb{Z}^3$ and (c) Λ_3 lattices, indicating (a) the first three shells of potential poll points on $\mathbb{Z}^3$ used in the LTMADS and OrthoMADS formulations, and (b, c) the first three shells of potential poll points used, respectively, in the $\mathbb{Z}$-MADS and Λ-MADS formulations. The number of points in all three sets of shells (arranged, respectively, in cubes, octahedra, and cuboctahedra around the CMP) are listed in Table 1.

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 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.

B. 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. Belitz & Bewley (2011) extend 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 con-

[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.

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

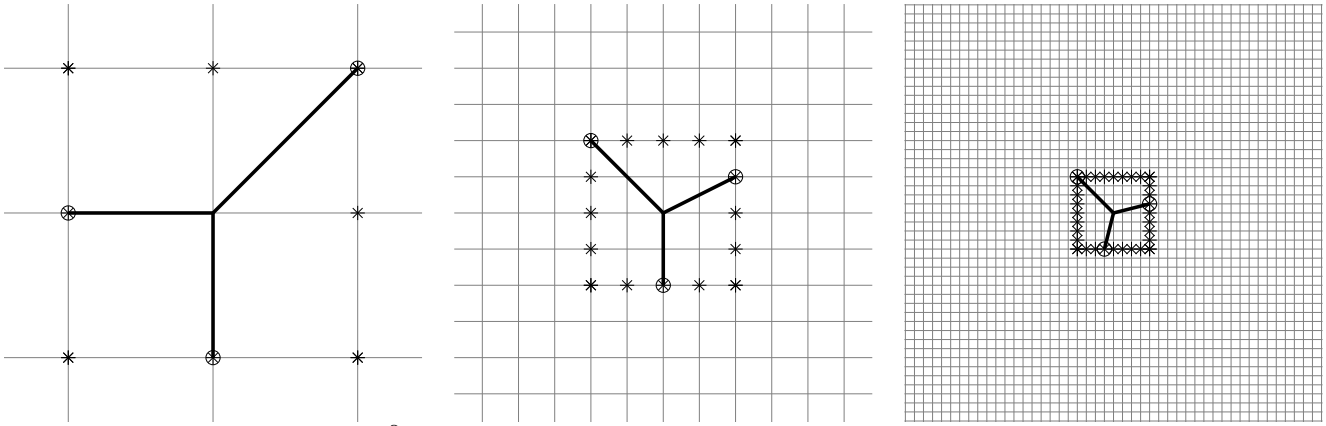

Fig. 3: The underlying Cartesian grid $\mathbb{Z}^2$ (——) and two successive factor-of-four refinements of this grid (from left to right) in the $n = 2$ LTMADS algorithm. Given a CMP at the center of each subfigure, the shell of points from which the poll points are selected are marked ($*$), and a representative poll set is indicated ($\circ$); this poll set forms a minimal positive basis (▬▬), with $n+1$ vectors around the CMP.

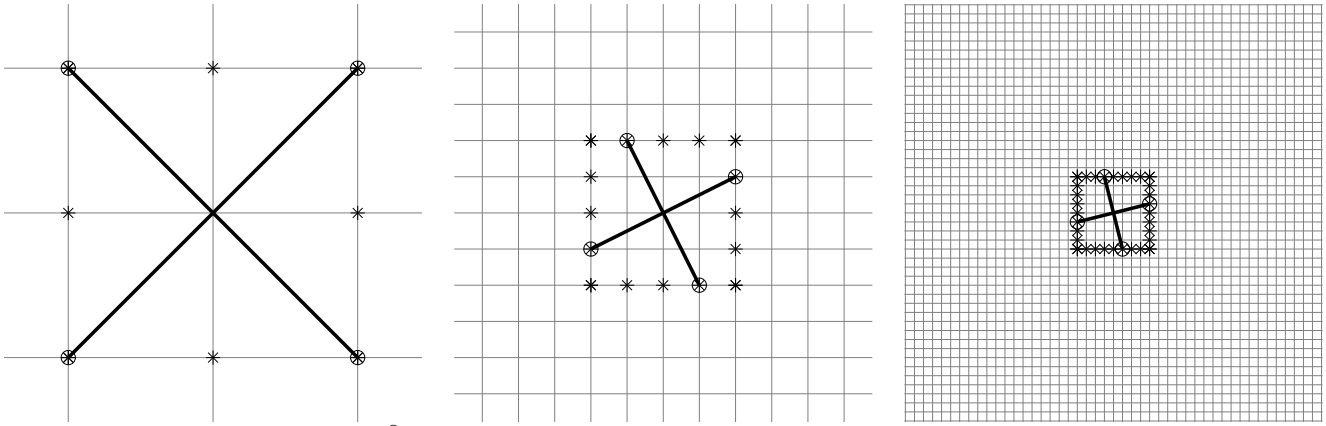

Fig. 4: The underlying Cartesian grid $\mathbb{Z}^2$ (——) and two successive factor-of-four refinements of this grid in the $n = 2$ OrthoMADS algorithm (cf. Figure 3). The shell of points from which the poll points are selected are marked ($*$), and a representative poll set is indicated ($\circ$); this poll set forms an (orthogonal) maximal positive basis (▬▬), with $2n$ vectors around the CMP.

strained, 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).

C. Mesh Adaptive Direct Search (LTMADS & OrthoMADS)

Mesh Adaptive Direct Search (MADS) algorithms are an alternative class of GPS 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 3) and OrthoMADS (Audet & Dennis 2008; for a graphical depiction, see Figure 4), the latter of which essentially supercedes the former, and is becoming increasingly popular for practical numerical optimization problems with expensive functions (see, e.g., Marsden et al., 2011). 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

[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.

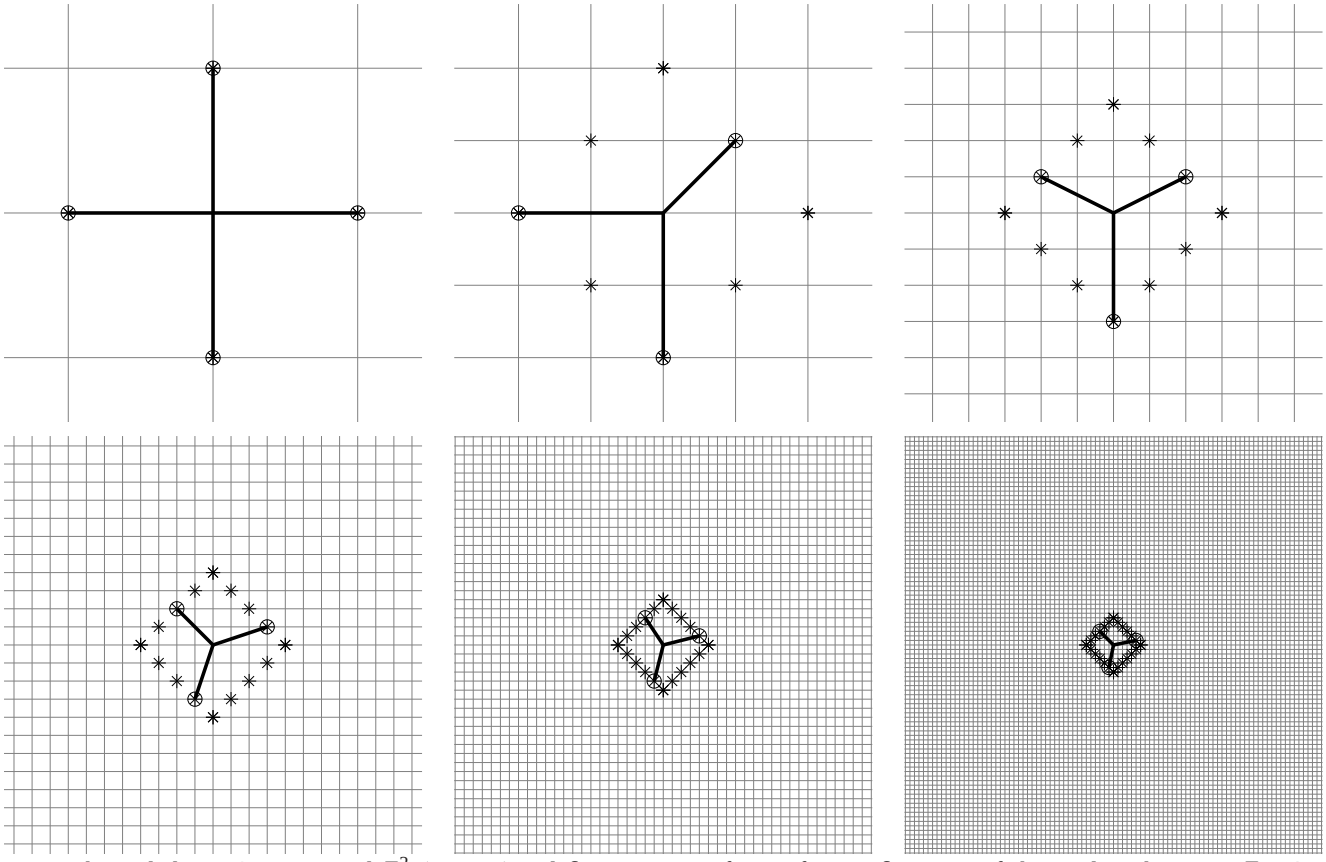

Fig. 5: The underlying Cartesian grid $\mathbb{Z}^2$ (———) and five successive factor-of-two refinements of this grid in the $n = 2$ $\mathbb{Z}$-MADS algorithm (cf. Figures 3 and 4). The set of points from which the poll points are selected are marked ($*$), and a representative poll set is indicated ($\circ$); 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 (▬▬).

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 1a and 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 3), whereas the OrthoMADS algorithm will select $2n$ of these points to poll (see Figure 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 1). Given an appropriate 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 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 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 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, 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

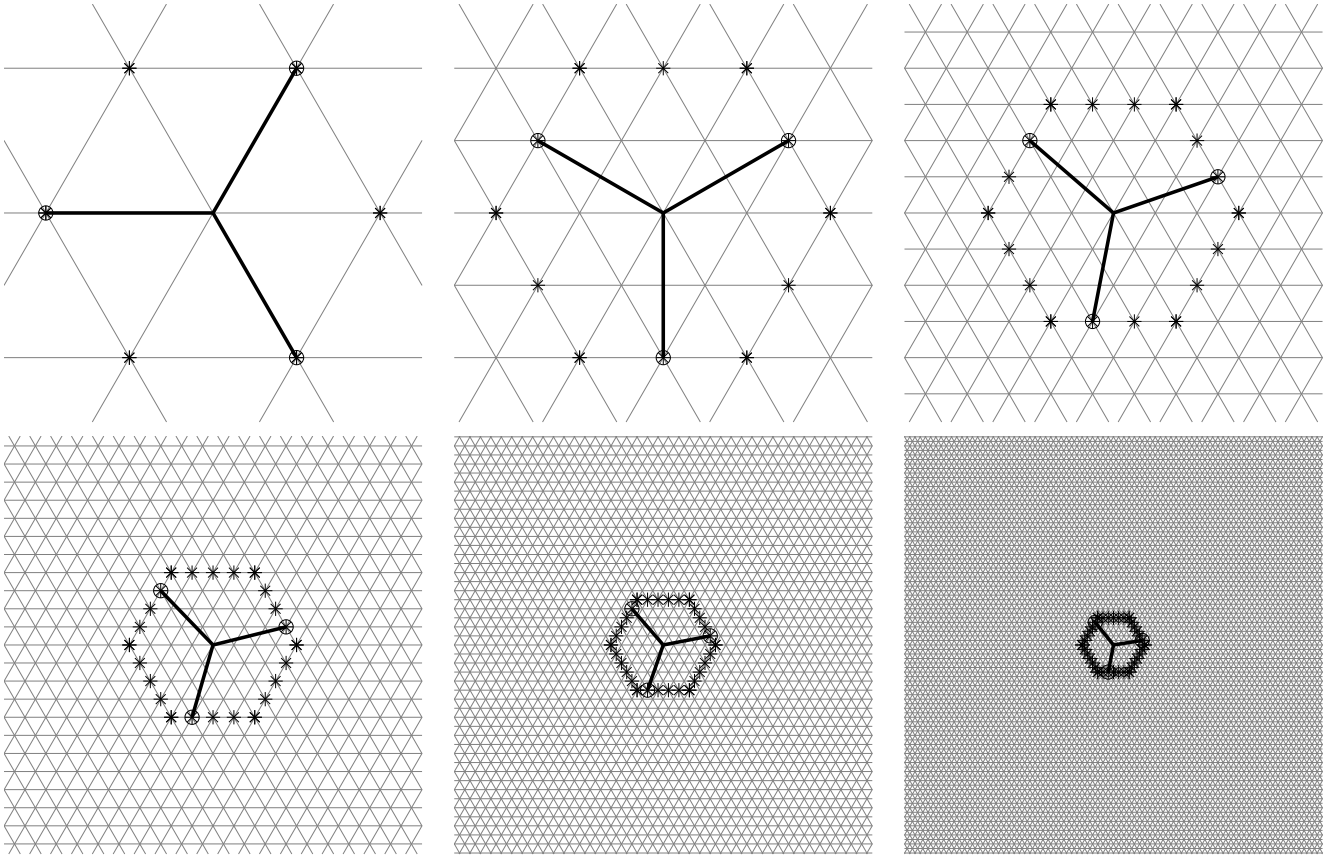

Fig. 6: The underlying Λ_2 lattice (——) and five successive factor-of-two refinements of this lattice in the $n = 2$ Λ-MADS algorithm (cf. Figures 3, 4, and 5). The set of points from which the poll points are selected are marked ($*$), and a representative poll set is indicated ($\circ$); this poll set forms a minimal positive basis around the CMP (▬▬).

to 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 requires $2n$ 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 increase in the number of function evaluations required for each complete poll step.

D. 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 3 and 4, is not the only choice for a MADS-type algorithm on a Cartesian grid. As illustrated in Figure 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 1b & 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 1). Noting the sentence at the end of the previous subsection, at each poll step, the preferred implementation of the $\mathbb{Z}$-MADS algorithm selects $n+1$ of these points to poll.

E. An overview of Lattice-based MADS (Λ-MADS)

The present paper 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

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

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 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:

$$
\begin{aligned}
S_{\Lambda_2}(x) &= (1+4x+x^2)/(1-x)^2, \\
S_{\Lambda_3}(x) &= (1+9x+9x^2+x^3)/(1-x)^3, \\
S_{\Lambda_4}(x) &= (1+20x+54x^2+20x^3+x^4)/(1-x)^4, \\
S_{\Lambda_5}(x) &= (1+35x+180x^2+180x^3+35x^4+x^5)/(1-x)^5, \\
S_{\Lambda_6}(x) &= (1+66x+645x^2+1384x^3+645x^4+66x^5+x^6)/(1-x)^6, \\
S_{\Lambda_7}(x) &= (1+119x+2037x^2+8211x^3+8787x^4+2037x^5+119x^6+x^7)/(1-x)^7, \\
S_{\Lambda_8}(x) &= (1+232x+7228x^2+55384x^3+133510x^4+107224x^5+24508x^6+232x^7+x^8)/(1-x)^8.
\end{aligned}
$$

Such series expansions are easily calculated in, e.g., Mathematica or Wolfram|Alpha; the first 10 terms of each of these series are listed in Table 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 §II, these implementation issues are addressed at length in §III. In §IX, we attempt to quantify the impact of each of the individual implementation issues discussed here and in §III 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.

II. 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 1a and 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 1b and 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

n	LTMADS/OrthoMADS shell shape	LTMADS/OrthoMADS radial nonuniformity	$\mathbb{Z}$-MADS shell shape	$\mathbb{Z}$-MADS radial nonuniformity	Λ-MADS shell shape	Λ-MADS radial nonuniformity
2	square	$\sqrt{2} \approx 1.414$	diamond	$\sqrt{2} \approx 1.414$	hexagon	$\sqrt{4/3} \approx 1.155$
3	cube	$\sqrt{3} \approx 1.732$	octahedron	$\sqrt{3} \approx 1.732$	cuboctaheron	$\sqrt{2} \approx 1.414$
4	tesseract	$\sqrt{4} = 2.000$	16-cell	$\sqrt{4} = 2.000$	24-cell	$\sqrt{2} \approx 1.414$
5	penteract	$\sqrt{5} \approx 2.236$	pentacross	$\sqrt{5} \approx 2.236$	(see text)	$\sqrt{5/2} \approx 1.581$
6	hexeract	$\sqrt{6} \approx 2.449$	hexacross	$\sqrt{6} \approx 2.449$	(see text)	$\sqrt{8/3} \approx 1.633$
7	hepteract	$\sqrt{7} \approx 2.646$	heptacross	$\sqrt{7} \approx 2.646$	(see text)	$\sqrt{3} \approx 1.732$
8	octeract	$\sqrt{8} \approx 2.828$	octacross	$\sqrt{8} \approx 2.828$	(see text)	$\sqrt{2} \approx 1.414$

TABLE II: 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.

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 1c and 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. 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].

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 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.

III. 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.

[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 1; thus, the radial nonuniformity values presented in Table 2 may be derived directly from the Θ and Δ values presented in Table 1.

[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.

A. 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 (see Bewley, Belitz, & Cessna 2011). 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 paper, Λ_2 through Λ_8, are given by

$$B_{\Lambda_2} = \begin{pmatrix} -1 & \\ 1 & -1 \\ & 1 \end{pmatrix}, \quad B_{\Lambda_3} = \begin{pmatrix} -1 & 1 & \\ -1 & -1 & 1 \\ & & -1 \end{pmatrix}, \quad B_{\Lambda_4} = \begin{pmatrix} -1 & 1 & & \\ -1 & -1 & 1 & \\ & & -1 & 1 \\ & & & -1 \end{pmatrix}, \quad B_{\Lambda_5} = \begin{pmatrix} -1 & 1 & & & \\ -1 & -1 & 1 & & \\ & & -1 & 1 & \\ & & & -1 & 1 \\ & & & & -1 \end{pmatrix},$$

$$B_{\Lambda_6} = \begin{pmatrix} & & & & & 1/2 \\ -1 & & & & & 1/2 \\ 1 & -1 & & & & 1/2 \\ & 1 & -1 & & & 1/2 \\ & & 1 & -1 & & -1/2 \\ & & & 1 & -1 & -1/2 \\ & & & & 1 & -1/2 \\ & & & & & -1/2 \end{pmatrix}, \quad B_{\Lambda_7} = \begin{pmatrix} -1 & & & & & & 1/2 \\ 1 & -1 & & & & & 1/2 \\ & 1 & -1 & & & & 1/2 \\ & & 1 & -1 & & & 1/2 \\ & & & 1 & -1 & & -1/2 \\ & & & & 1 & -1 & -1/2 \\ & & & & & 1 & -1/2 \\ & & & & & & -1/2 \end{pmatrix},$$

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

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 1. Associated with each of these lattices is a straightforward and computationally efficient *quantization* algorithm, described in §5 of Bewley, Belitz, & Cessna (2011), which takes any point in $\mathbb{R}^n$ and computes the closest point on the discrete lattice Λ_n.

1) 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 1), then finish, otherwise, increase *m* by one and try again.

2) 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 1c & 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 1c & 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.

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

[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 2, the average radius of each target shell can be well approximated quite easily.

B. 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.

C. Refining the mesh

As mentioned previously and illustrated in Figure 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 1c & 2c and Figure 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 2 of §II. As a consequence, the radial uniformity of the Λ-MADS poll sets is substantially better than the radial uniformity of the LTMADS/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 §III-D.4). As listed in Table 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 curiousity.

Thus, in addition to a *coarsest* grid spacing to be used by the optimization algorithm (see the first paragraph of §III and footnote 14), 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 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.

D. Generating new poll sets

1) 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 §III-D.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.

[17] As mentioned in §III-A.2, this method is modified slightly in the practical Λ-MADS algorithm for the outer shells.

[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 1) does not actually benefit the overall rate of convergence of the practical LTMADS or OrthoMADS algorithms.

2) 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). 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 §III-A.2. 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). If points so generated do not positively span the 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.

3) 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.

4) 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 §III-D.2: "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 previously; 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 span the neighborhood of the CMP) can often be reduced significantly.

5) Reorienting the poll set in a low-discrepency fashion after one or more unsuccessful poll steps:

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 generates a new poll set which is not only itself highly uniform, but is also generally well differentiated from the directions of the

[19] We may initially take $p = n+1$; note that this algorithm is easily and naturally extended in three important ways in §III-D.4, §III-D.5, and §III-D.6.

[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 set that these additional fixed particles might otherwise create.

previous (failed) poll sets around the CMP, thus generalizing naturally the idea of low-discrepancy *sequences* of vectors to low-discrepancy *sets* of vectors.

6) Optional step: including a poll point designed to accelerate convergence when the function is locally C^1: As discussed in the second paragraph of §I-A, 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 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.

E. 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 §III-B), 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 §III-F, 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.

F. Coarsening the mesh

Still need to write this subsection, in the style of the above.

IV. 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

[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.

n		2	3	4	5	6	7	8
p	Shells $1,2,4,8,..$ vs Shells $1,2,3,4,...$	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

TABLE III: Convergence comparison of fundamental features of the Λ-MADS algorithm

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 III, 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 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 III, 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 III. 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 III. 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 III. 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.

n	2	3	4	5	6	7	8
p	57.1	55.26	55.15	54.9	56.1	56.6	54.4
r	0.90	0.923	0.9421	0.8631	0.8758	0.8757	0.901

TABLE IV: Performance comparison between OrthoMADS and Λ-MADS on the Rosenbrock test function.

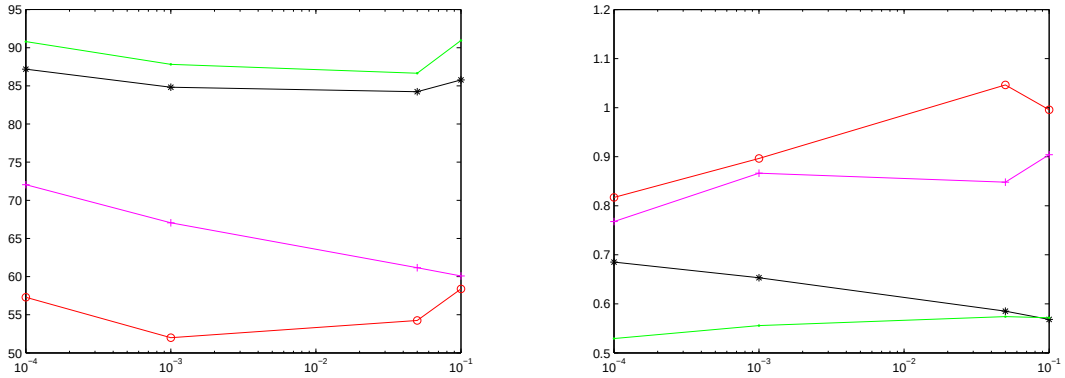

Fig. 7: The efficiency metrics p (L) and r (R) comparing Λ-MADS to OrthoMADS on the Rosenbrock cost function for convergence goals of $0.1, 0.05, 0.001, 0.0001$ in dimensions $n = 2$ (red circles), $n = 4$ (magenta $+$), $n = 6$ (black asterisk), and $n = 8$ (green dots). Note how Λ-MADS outperforms OrthoMADS in high dimensions for all convergence levels.

V. 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

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

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 V. 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 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.

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 V.

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

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

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.

VI. 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 Bewley, Belitz, & Cessna (2011)). 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 2.

In the interest of remaining computationally feasible, the constraints must be 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 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, & Cessna, (2011).

Figure 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 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.

VII. 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 careful 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 refine the mesh by a factor of 2 rather than a factor of 4, (4) to utilize a minimal rather than a maximal positive basis in the appropriate dimensions, and (5) to implement incomplete polling of the Poll sets. In dimensions $n = 2,3,4$ a maximal positive basis provides superior convergence to a minimal basis. In addition, Λ-MADS incorporates a mesh coarsening scheme that allows effective poll orientations to be maintained on the coarser mesh,

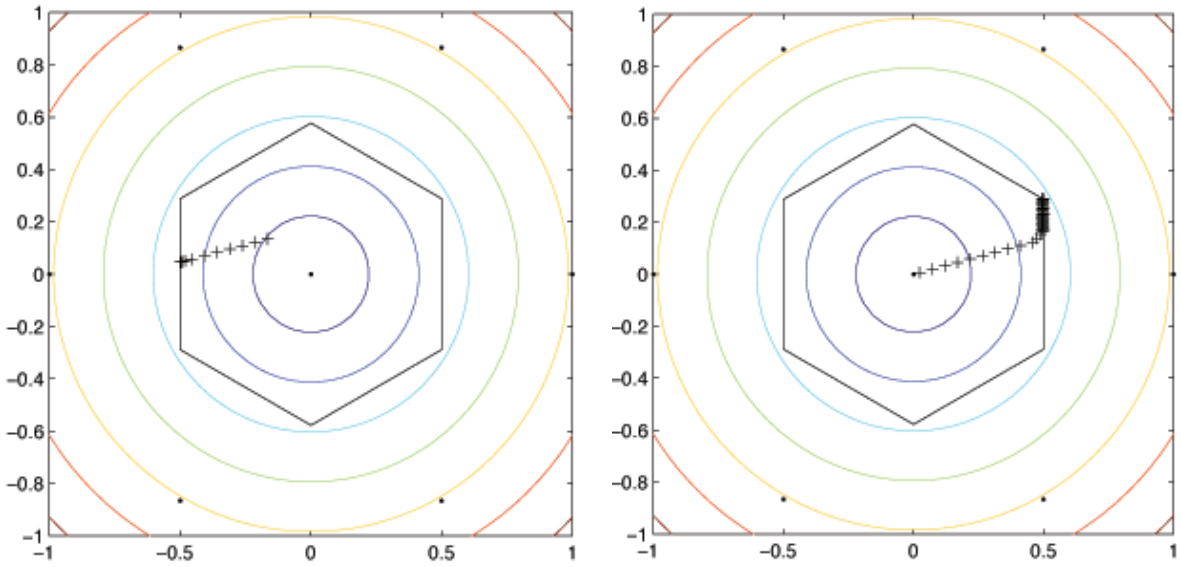

Fig. 8: Locating the deep holes of the A_2 lattice utilizing Successive Polling (left) and Λ-MADS (right). The hard constraints are indicated in black; the cost funtion contours are plotted as well. The inability of the SP algorithm to handle constraints prevents convergence; Λ-MADS maintains convergence and locates the deep hole.

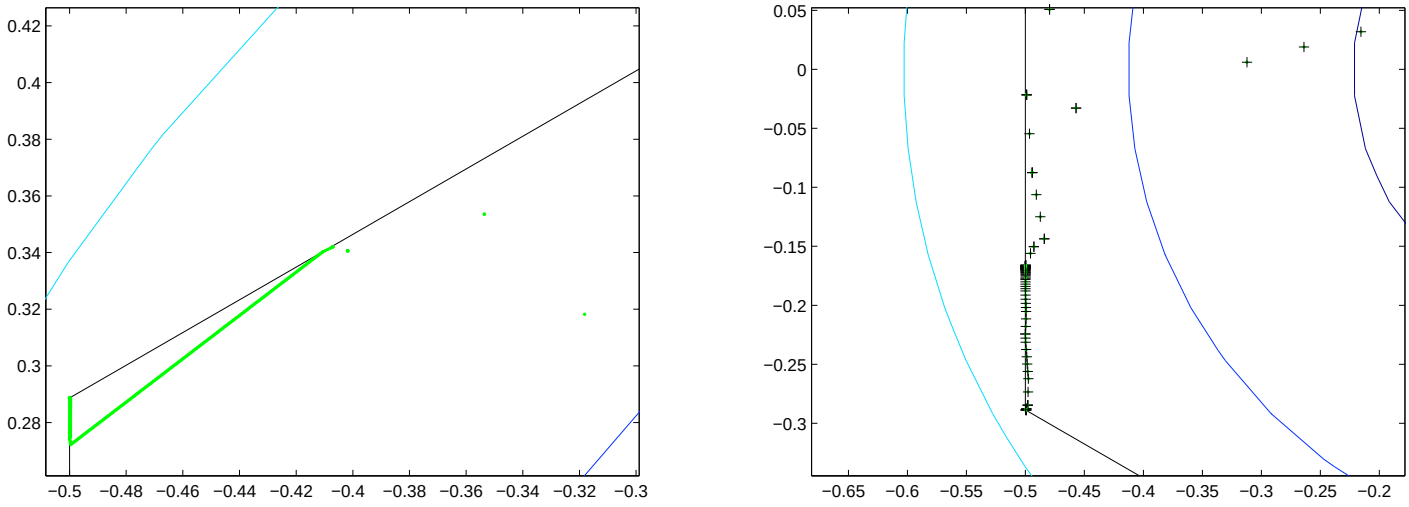

Fig. 9: Locating the deep holes of the A_2 lattice utilizing Λ-MADS with no coarsening, plotting the CMP as green dots (L), and coarsening implemented (R), plotting the CMP as a black cross. The algorithm with coarsening enabled clearly maintains a larger average step size, speeding convergence in the presence of active constraints.

preventing poor coarsening behaviors evident in other algorithms. Unlike most GPS algorithms, Λ-MADS can handle hard constraints, as effectively demonstrated on the example research application of locating the deep holes of a lattice.

Testing of Λ-MADS against the competing OrthoMADS algorithm on quadratic bowls and the *n*-dimensional Rosenbrock function indicates that Λ-MADS generally converges more rapidly than OrthoMADS, with the performance difference becoming greater in higher dimensions. Combining the good convergence efficiency demonstrated in these tests and the algorithm structure that inherits all the convergence behaviors of previous MADS algorithms, Λ-MADS is the most efficient MADS algorithm yet developed, and is the clear choice for difficult modern convex optimization problems.

* OLD STUFF IS INCLUDED BELOW THIS LINE. PULL ABOVE THIS LINE AND MASSAGE AS NECESSARY. *

In the second case, a minimal positive basis consisting of $N+1$ points is desired. To locate the optimal minimal positive basis on S_k, defined as the most spatially regular basis, a force-based optimization of N points on the hypersphere is performed, holding $\hat{\mathbf{d}}$ fixed, as described in Belitz & Bewley (2011). Specifically, each of the $N+1$ points are treated as charged particles on a hypersphere, such that each exerts a force on the others proportional to the inverse of the Euclidean distance separating them. Keeping the location of $\hat{\mathbf{d}}$ fixed, the force on each of the remaining particles is calculated. Then, the location of the N points on the hypersphere is iteratively updated to minimize the greatest force experienced by any of the points. The points reach equilibrium forming a perfectly symmetric minimal positive basis by definition. Then, the Poll Set is calculated, as above, by finding the points in S_k that best approximate these ideal directions. As above, upon refinement of the Shells, the possible directions of the poll set becomes dense, and the ideal directions generated via the force-based optimization become included in the Shell.

A. Choosing a Polling Direction

In both variats of Λ-MADS described above, an initial polling direction $\mathbf{d}$ is necessary to build a poll set on the shell S_k. In LTMADS, the first poll vector is located by taking a normally distributed random direction in the hypersphere, then restricting to the allowable fine mesh points. OrthoMADS adds a deterministic component to the direction by introducing a poll vector based on a Halton Sequence. The Halton Sequence offers a pseudo-random, repeatable number sequence that gives points that become dense in the unit hypercube. By multiplying by 2 and shifting by the vector of ones, the Halton Sequence easily gives vectors that become dense in the space

$$\mathbf{x} \subset R_n \; s.t. \; ||\mathbf{x}||_\infty = 1$$

The advantage of using a Halton Sequence based polling direction is that the sequence is repeatable, and large variations from iteration to iteration are avoided. The repeatability is important when the polling direction is to be reused as previously described, being updated only when necessary. However, a Halton direction is suboptimally posed in an efficiency context, particularly when a MADS algorithm is implemented on a lattice.

To maximize efficiency, the entire poll must be as uniform as possible, both with respect to angular and spatial symmetric of a single poll set, as well as minimizing angular and spatial irregularity from poll set to poll set, on a given grid. This necessity leads directly to the interest in the application of lattices to the MADS algorithm. The Halton based polling direction is less uniform than it could be in an angular uniformity sense, as it biases the initial poll direction toward a corner of the cube in which the Halton direction is defined. In Figure 3 below, the angle between a given initial poll direction and the x-axis was calculated $50,000$ times, with the results summarized via a histogram. As $N=2$ in this case, it it clear that the four corners of the unit cube are overproportionally represented. A random vector dense on the hypersphere, however, seen in Figure 3 as well, is clearly uniform, and does not bias the poll set. The bias toward certain polling directions will limit the performance of the MADS algorithm, as a good descent direction will be located less regularly. When the direction is dense in the hypersphere, the projection of the direction onto the feasible Cartesian Shell in OrthoMADS will be uniformly dense, but once the Shell begins to approximate a hypersphere, the direction should be chosen in a fashion to be dense in a hypersphere rather than hypercube.

The Halton sequence chosen in OrthoMADS becomes dense in a hypercube, meaning that the polling directions on the faces of the allowable hypercube are equally dense. However, when the allowable region moves from a hypercube to a hypersphere, the Halton direction used in OrthoMADS will bias the poll set.

To demonstrate the effect, 9000 optimization runs were performed in $N=10$, using the minimal basis Λ-MADS algorithm described below. Randomly permuted quadratic cost functions were shifted to a random minimum location. Two versions of Λ-MADS were considered: one utilizing a Halton sequence generator, the other a random direction in the hypersphere. Direction re-using and incomplete polling as discussed below were implemented for these tests. Both algorithms were started on the same grid at the same point, and the average number of function evaluations necessary for the same level of convergence was computed. The algorithm utilizing a hypersphere measureably outperformed its competitor, requiring, on average, 765 function evaluations, compared to 778 for the Halton-based algorithm. The performance hinderance induced by the bias of the direction dense in the hypercube is clearly measurable. For greatest efficiency, the polling direction should clearly be chosen via a method that produces directions uniformly dense on the surface of the hypersphere. This can be as simple as simply choosing a random point on the hypersphere, utilizing only the Halton directions that lie within the hypersphere, or any other method that produces a uniform distribution. The Λ-MADS algorithm presented herein simply utilizes a uniformly-distributed random direction.

B. The Complete Λ-MADS Algorithm

By utilizing a very efficient polling direction as described above (a sequence dense inside the unit hypersphere), combined with a basis located on an efficient lattice via the algorithms described in Section 5, a new MADS algorithm is fully defined. Any lattice can be chosen for this purpose, though lattices with greater packing density, kissing number, and coordination sequences should provide shells more closely approximating a sphere, and thus should provide a more uniform, and thus

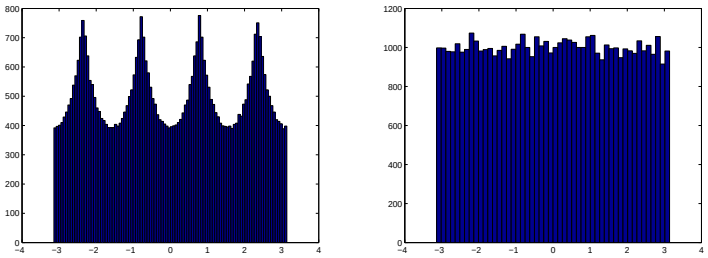

Fig. 10: Histograms showing the angle between the first polling direction and the x-axis, for a Halton-based sequence (L) and a random vector on the hypersphere (R).

more efficient, MADS algorithm. For low dimensions, the A_n, D_n, and E_8 lattices are good choices, and are thus chosen for numerical testing. In high dimensions the laminated lattices discussed in Bewley, Belitz, & Cessna (2011), would be the logical choice, as these offer the greatest coordination sequences of all lattices. For brevity we point the reader at Belitz, Bewley, & Cessna (2011), for the basis matrices and quantization algorithms for a great number of suitable lattices.

The nearest neighbors of the lattice, that is, the shell S_1 of all points one hop from the origin, can be numerically enumerated by multiplying the basis matrix by an exhaustive sequence of integer combinations contained in a vector, until all the nearest neighbors in S_1 have been determined, precisely as discussed in Belitz & Bewley (2011).

With S_1 defined, and the ability to quantize a random initial CMP onto the lattice via the algorithms summarized in Bewley, Belitz & Cessna, (2011), a MADS algorithm can be implemented as discussed above.

C. Extension To High Dimensions

Consider the augmentation of the algorithm above, in that the subset of lattice points that can be used to form a poll set are redefined. Given a coarse mesh size Δ_c and a fine mesh size Δ_f as discussed above, by definition the distance between neighboring points on the fine mesh is Δ_f and the distance between neighbors on the coarse grid is Δ_c. The aim of MADS is to locate a set of poll points that lie on the fine grid, whose distance from the CMP is bounded by the coarse grid size. This can accomplished by enumerating shells of the lattice as above, but can also be performed by locating a perfectly distributed minimal basis defined by a size Δ_c and the poll orientation. Then, rather than restricting to the appropriate Shell points, the poll set is identified simply by restricting the minimal basis to the fine lattice.

This is a slight modification from the previous algorithm in that the allowable fine mesh points are not bounded by a hop count, rather a Euclidean distance defined by Δ_c and the deep hole radius of the fine grid r_{dh}. From there, a basis is located as described above, and for each poll direction a poll point is located to bound the distance from the CMP by $R+R_{dh}$. Thus, each poll point is a maximum distance from the CMP while being bounded above by the coarse mesh size. When the coarse mesh and the fine mesh sizes is small, the set of points this algorithm makes available for the poll set reduced to exactly the shells defined above. After many refinements, fine mesh points lying on a shell further from the CMP will be included. As this algorithm requires only restriction to the fine lattice, there is no overhead induced by the need to enumerate shells in their entirety, and thus scales to high dimensions efficiently. Additionally, it improves even further upon the uniformity of the lattice shells.

The polling direction is calculated via a random number dense in the hypersphere, a pseudo-orthogonal or minimal positive basis can be located and restricted to the Shell.

VIII. Isolated numerical testing of each component issue

A. The effect of the lattice

To quantify the behavior of implementing a maximal positive basis in Λ-MADS and comparing to OrthoMADS with a random polling direction utilized gives a quantification of simply replacing the underlying lattice in a MADS optimization. To this end, a statistically relevant number of optimizations with each algorithm were run, optimizing randomly selected quadratic functions, with a randomly selected initial CMP and randomly selected function minimum. Each cost function $F(x)$ is defined as follows:

$$F(x) = (x-\bar{x})A(x-\bar{x})^T$$

Where $\bar{x}$ is a randomly distributed vector, and A is a random positive definite matrix. Both Λ-MADS and OrthoMADS were allowed to converge to 0.1% of the initial cost function value. The lattices were scaled to start with constant initial vornoi cell volume of the A_n lattice and the cartesian grid. Table 2 below summarizes the effect of redefining OrthoMADS on a lattice. The two parameters quantifying performance are the percent of total runs that Λ-MADS converged faster than

n	2	3	4	5	6	7	8
p	57.1	55.26	55.15	54.9	56.1	56.6	54.4
r	0.90	0.923	0.9421	0.8631	0.8758	0.8757	0.901

Table 2. Performance comparison between the D_n-based Λ-MADS algorithm and the $\mathbb{Z}^n$-based OrthoMADS algorithm applied to randomly shifted quadratic bowls. Notice how in the relatively moderate dimension of $n = 8$, Λ-MADS requires approximately 30% as many function evaluations to converge compared to OrthoMADS.

n	2	3	4	5	6	7	8
p	46.15	53.14	58.93	72.49	76.92	79.92	86.02
r	1.145	1.399	0.9642	0.7240	0.5653	0.4777	0.3806

Table 3. Performance comparison between the D_n-based maximal Λ-MADS algorithm and the D_n-based minimal Λ-MADS algorithm applied to randomly shifted quadratic bowls. For $n > 3$, it is seen that the minimal basis algorithm converges faster than the maximal basis algorithm. This validates the suggestion often made in the literature that a minimal positive basis is the most efficient choice for unconstrained optimization.

n	2	3	4	5	6	7	8
p	55.6	57.0	59.8	59.0	64.32	69.0	75.43
r	0.835	0.869	0.8687	0.9054	0.8644	0.8466	0.7824

Table 4. Performance comparison between the slow refinement minimal Λ-MADS algorithm and the standard refinement Λ-MADS algorithm applied to randomly shifted quadratic bowls. In all dimensions, the slower refinement scheme results in superior convergence behavior.

Orthomads, p, and the ratio of the average number of functional evaluations Λ-MADS required to the number of evaluations that OrthoMADS required.

As can clearly be seen in Table 2, the effect of modifying to an efficient underlying lattice while maintaining the same basis type (orthogonal and maximal) sees a measurable improvement compared to Cartesian; however, the effect is small and proved sensitive to initial conditions.

B. Minimal versus Maximal Positive Basis

In the formulation of Λ-MADS above, to variants are presented: one utlizing a maximal positive basis that becomes asymptotically orthogonal, and a minimal positive basis that optimizes for angular and radial uniformity on the Shell. Historically, various authors have suggested that a minimal basis should provide a higher convergence rate, as fewer points are evaluated per poll step. As both options are available in the Λ-MADS, a series of tests were performed to capture a statistically relevant measure of convergence performance. The same quantification of convergence was performed as described above, and incomplete polling was implemented. The results are summarized in Table III below.

As the results in Table 3 illustrate that, when given the option between a well-distributed minimal basis and a well-distributed maximal basis, the minimal basis performs significantly better, as evidenced by requiring only 38% as many function evaluations as the maximal algorithm.

C. Grid refinement options

All variants on Successive Polling algorithms refine the underlying grid by a factor of 2 upon an unsuccessful Poll step. The factor of 2 has historically been chosen as there is no smaller factor to refine by that will allow all previous function evaluations to be located on valid lattice nodes independently of the eventual lattice scale. Moving to a MADS framework allows for contemplation of the refinement factor.

Previous MADS implementations refine the coarse and fine grids by factors of 2 and 4, respectively, which, like the SP algorithms mentioned above, allows all previous function evaluations to lie on the grid at all levels of refinement. Upon inspection we see that Λ-MADS refines the fine lattice scale by a factor of 4, and by refining the coarse grid by a factor of 2, the shells upon which the positive basis lie increase as 2^k, where k is the number of refinements performed. Clearly, this is not the only factor by which one can refine while keeping all function evaluations on the fine lattice. If the fine refinement factor is chosen to be 2. At the k'th refinement, the coarse grid scale is $k\Delta_f$. Thus, the size of the poll set refines more slowly than if the normal factor of 2 were used.

The effect of this slower refinement scheme is an increase in convergence rate. The algorithm (1) refines less quickly while searching for an appropriate poll set orientation, thereby allowing larger steps toward the function minimum once a suitable orientation has been located, and (2) generally prevents the algorithm from overrefining when matching the local terrain characteristics of the cost function. To demonstrate, the same numerical analysis described above was performed comparing two Λ-MADS algorithms. Both utilize a minimal basis, incomplete polling, however, one refines the mesh by 4 (as LTMADS and OrthoMADS do); the other refines by a factor of 2 as decribed above. Table 4 below summarizes the results.

The numerical test demonstrate that the smaller refinement factor leads to an increase in convergence rate, requiring only 78% as many function evaluations to convergence in $n = 8$.

n	2	3	4	5	6	7	8
p	64.5	48.82	60.2	76.00	83.10	89.63	93.17
r	1.21	1.46	0.9235	0.6002	0.4291	0.3555	0.3034

Table 5. Performance comparison between the D_n-based Λ-MADS algorithm and the $\mathbb{Z}^n$-based OrthoMADS algorithm applied to randomly shifted quadratic bowls. Notice how in the relatively moderate dimension of $n = 8$, Λ-MADS requires approximately 30% as many function evaluations to converge compared to OrthoMADS.

n	2	3	4	5	6	7	8
p	78.58	68.96	72.8	76.00	83.10	89.63	93.17
r	0.8026	1.165	0.8596	0.6002	0.4291	0.3555	0.3034

Table 6. Performance comparison between the slow refinement Λ-MADS algorithm and the $\mathbb{Z}^n$-based OrthoMADS algorithm applied to randomly shifted quadratic bowls. Notice how in the relatively moderate dimension of $n = 8$, Λ-MADS requires approximately 30% as many function evaluations to converge compared to OrthoMADS.

IX. Numerical Tests

As established in Belitz & Bewley (2011), GPS algorithms display superior convergence rates under maximally uniform poll sets. Efficient lattices provide a superior set of points available (as indicated by the Coordination Sequences of these lattices) to be selected as Poll points compared to the ubiquitous Cartesian grid. Thus, running GPS algorithms on non-Cartesian lattices tends to significantly improve convergence rates on typical cost functions, with the performance increase becoming greater as the dimension of the problem increases. We hypothesize that the same behavior should be evident in a MADS algorithm, where an efficient lattice provides a more uniform poll set from iteration to iteration. To test this hypothesis, following the convergence analysis of lattice-based GPS algorithms in Belitz & Bewley (2011), the D_n-based MADS algorithm Λ-MADS utilizing a minimal positive basis was tested against OrthoMADS on a statistically relevant number of randomly shifted quadratic cost functions.

For these tests, Λ-MADS utilizes a minimal positive basis; both algorithms use incomplete polling (that is, the Poll step is terminated as soon as an improved CMP is located). For purposes of comparison, neither algorithm uses a Halton Sequence generated polling direction, instead using random polling directions dense in a hypersphere, and coarsening was not implemented. The two optimization codes tested are clone codes; the only difference between them is the poll set generated.

Again, the MADS optimizations were started at the same randomly selected CMP, and then the number of function evaluations necessary to converge to a given level of convergence was recorded. Two metrics of performance are considered: the average normalized number of function evaluations necessary to converge n, and the frequency that Λ-MADS converged with fewer evaluations than OrthoMADS r. The results are tabulated in Table 5.

There is significant subtlety in selecting an accurate and relevant comparison of efficiency between two related optimization algorithms. Great pains were taken to make the results reported above as meaningful as possible. The point of possible contention rests in the initial scaling of the lattice or grid during the numerical analysis above. If the initial grid scaling has a very coarse Cartesian grid and a very fine lattice scale, and the distance from the original CMP to the function minimum is much greater than either grid scale, the Cartesian code will appear to be more efficient. If the scalings are reversed, with the lattice being coarse and the grid being fine, then the lattice will appear to be radically more efficient than the grid. Neither of these results is legitimate.

To make all comparisons meaningful, all lattices are scaled such that the voronoi cell volume of the lattices are constant at the beginning of the optimizations. Secondly, the grid scale was selected to be within one order of magnitude as the distance between the random CMP and random function minimum. Our selection of these parameters is to accurately represent a reasonable decision of lattice scaling if the optimization were being performed on a very poorly understood function. When utilizing these algorithms in a research application, it is preferable to choose an initial lattice scale on the length scale of the feasible space, thereby keeping function evaluations far apart until convergence is approached.

As is clear from Table 4, the D_n lattice-based Λ-MADS massively outperforms OrthoMADS. Clearly, on well-behaved functions, an efficient lattice combined with a minimal positive basis allows greater efficiency compared to the Cartesian grid. As in a GPS search, the performance difference can be explained as a combination of uniformity of the poll set and poll set size. Analogously to the SP algorithm being less efficient at finding a descent direction (Belitz & Bewley, 2011) on a minimal positive Cartesian basis, OrthoMADS produces polling vectors that range from normalized length 1 to length $\sqrt{(N)}$. When the cost function is challenging OrthoMADS will have to refine the grid size until the largest Poll sets are sufficiently refined to realize a descent step, thus limiting the performance of the smallest Poll sets. The more uniform the Poll set is from Poll to Poll, the more efficiently the algorithm can step toward the function minimum.

To demonstrate the effect of moving from a maximal positive basis to a minimal positive basis, negating any effects from lattice scalings or poll set uniformity, an identical comparison was made between Λ-MADS with a maximal orthogonal basis and Λ-MADS with a minimal positive basis. Again, the test function was a quadratic bowl with a random function

n	2	3	4	5	6
p	52.3	60.4	58.1	72.49	71.71
r	0.06527	0.1325	0.1426	0.04715	0.04362

Table 6. Performance comparison between the D_n-based minimal Λ-MADS algorithm and the OrthoMADS algorithm applied to the well-known Rosenbrock function, shifted such that the function minimum lies at a random location. Again, Λ-MADS convincingly outperforms OrthoMADS in moderate to high dimensions.

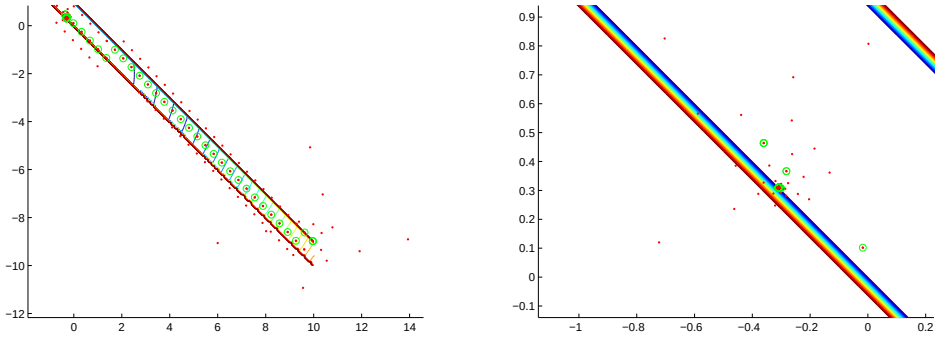

Fig. 11: Λ-MADS convergence behavior with hard constraints. The function minimum lies at $(-0.4934, 0.4934)$. At every iteration the complete Poll Set is plotted in red; the CMP of each iteration in green. Note poll direction reusing (left), giving GPS-like convergence. Once the CMP is near the function minimum the polls are generally unsuccessful, and it is clear how the polling directions become dense, as seen in the zoomed-in figure on the right.

minimum, a random initial CMP, but this time the D_n lattice was used for both algorithms and both lattices were identically scaled. The results can be found in Table 5. The test codes are clones of one another, with the sole difference between them being the number of poll points used to build a positive basis. The lattices were scaled identically, starting at the same CMP, and converging to the same values. Incomplete polling was implemented, and function evaluations were reused (that is, if a member of the poll set had been previously evaluated, the cost function was not called a second time). The performance difference by simply using a well-selected (that is, uniform) positive basis is significant.

These computational results indicates that Λ-MADS has an advantage over OrthoMADS in its ability to utilize a well structured minimal basis, while not sacrificing any convergence characteristics, further discussed below.

Optimizing a quadratic cost function does not capture all relevant behavior. Further testing, identical to the above, but on the n-dimensional Rosenbrock function was performed, again comparing Λ-MADS with a minimal basis to OrthoMADS. These results can be seen in Table 6. As predicted, Λ-MADS has superior convergence rates to OrthoMADS.

X. Constraint Handling

The primary advantage that MADS algorithms have over simpler GPS algorithms is MADS' ability to converge in the presence of hard constraints. When constraints are handled as a differentiable penalty function, a GPS algorithm will eventually successfully converge. However, if constraints are hard, a GPS search will generally stall, as the poll cannot achieve a descent direction. Various methods attempting to resolve this limitation have been explored. For example, a more sophisticated penalty method known as the Progressive Barrier approach can be effective. In the presence of very simple upper and lower bounds on the parameter space, a maximal orthogonal basis can be used at the constraints (see Booker, et al (1999). However, there are many engineering problems where the boundary surfaces are nonlinear, complex, or even unknown. A good MADS algorithm, correctly implemented with a carefully selected poll orientation selection algorithm, will successfully converge under constraint conditions which foil simple GPS algorithms. Given the ability of MADS algorithms to generate a dense set of poll points as the grids are refined, the Poll set will eventually contain a descent direction. Be reusing the successful poll direction, MADS can stochastically locate a descent direction and then proceeds to reuse that direction until a new CMP is not located, and new polling directions are explored. Note that the reusing of the polling direction is essential to allow convergence under hard constraints. If a new poll set direction is selected at every poll step MADS algorithms will not perform significantly differently from Successive Polling.

The effect of the poll direction reusing can be seen in Figure 7. The cost function is $F(\mathbf{x}) = \mathbf{x}^2$ subject to $x_1 + x_2 > 0$ and $x_1 + x_2 < 1$. Originally, the coarse mesh is too coarse to locate a superior CMP. Upon refinement, a descent direction is located, and that direction is reused. Once the algorithm approaches convergence, that direction fails to locate a new CMP, and new poll directions are generated.

Λ-MADS is a direct analogue to OrthoMADS, and thus shares all the constraint-handling behaviors of OrthoMADS. As the meshes refine, the angular uniformity of the positive basis quickly nears 1, and the directions explored become dense.

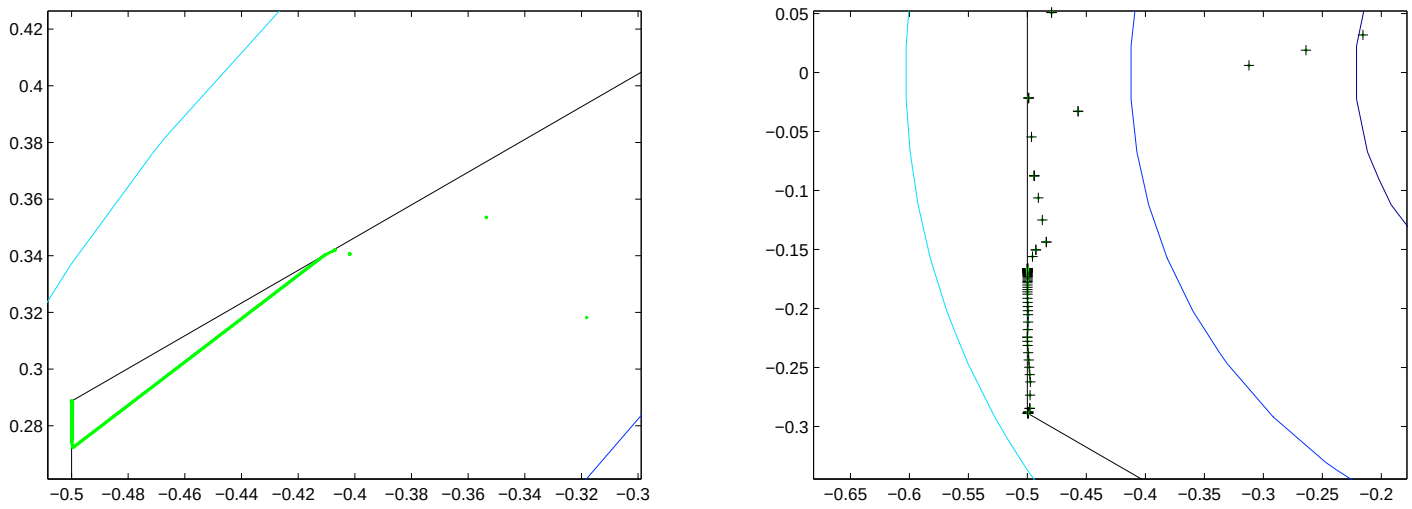

Fig. 12: Locating the deep holes of the A_2 lattice utilizing Λ-MADS with no coarsening, plotting the CMP as green dots (L), and coarsening implemented (R), plotting the CMP as a black cross. The algorithm with coarsening enabled clearly maintains a larger average step size, speeding convergence.

Both OrthoMADS and Λ-MADS update the poll direction following an unsuccessful Poll step, reusing the poll direction when the previous step was successful. Thus, Λ-MADS inherits OrthoMADS' behavior in constrained optimization, while significantly improving upon OrthoMADS' convergence rates on unconstrained cost functions.

In practice, there are many augmentations of a GPS or MADS algorithm that speed convergence. Some of these augmentations include using a surrogate function fitted to previously evaluated points to determine in what order to evaluate the cost function on the poll set (such as implemented in the LABDOGS algorithm and discussed in Belitz & Bewley (2011)), maximizing the likelihood of terminating the poll step upon finding a superior point. Another idea, the 'optimist's strategy', consists of reusing the successful poll vector of each successful Poll step, shifted to the new CMP, to give the location of a new function evaluation, based on the optimistic strategy that if the direction provided a descent once, it is likely to provide another. However, all these strategies are independent algorithms, such as LABDOGS, and are thus not considered here. In practice, careful consideration of Search algorithms to pair with any MADS polling must be taken, as potential rewards are undeniable.

XI. Mesh Coarsening

When Λ-MADS is implemented as described above, the algorithm convergences in the presence of hard constraints. When a constraint is active, the algorithm iteratively locates a descent direction (nearly) parallel to the constraint, allowing movement along the constraint. Of course, in the process of locating a feasible poll orientation, the mesh size is often refined greatly, and the resulting convergence is slow. This behavior is easy to understand upon inspection of Figure X. The motivation in coarsening the mesh size in addition to refining is the desire to stochastically locate a good poll orientation, and then take the largest feasible steps utilizing that orientation, minimizing the number of function evaluations required for convergence. OrthoMADS implements coarsening at every successful poll step, decrementing the Halton Sequence and increasing the fine mesh size and the coarse mesh size by a factor of four and two, respectively. The limitation of this approach is that the successful poll orientation is not reused upon coarsening. Thus, the coarsened poll step is biased to failure, as its poll orientation has already been determined to be suboptimal for that region.

Instead, the goal is to locate an effective poll orientation on the fine grid, then reuse that orientation on a coarse grid when possible. First, it is necessary to give a definition of an acceptable poll direction: a poll direction that gives m or greater consecutive successful poll steps, where $m > 1$. Upon the location of an acceptable poll direction, (that is, after m consecutive successful poll steps), the fine grid and hop count are decremented by a factor of 2 and 1, respectively. The poll direction on the coarsened mesh is the acceptable direction identified previously. If the poll step on the coarsened mesh is not successful, the mesh and hop count are refined as normal; however, a new poll direction is not selected. If the poll step on the coarsened mesh is successful, the algorithm continues under the poll direction selection and refinement criteria previously discussed.

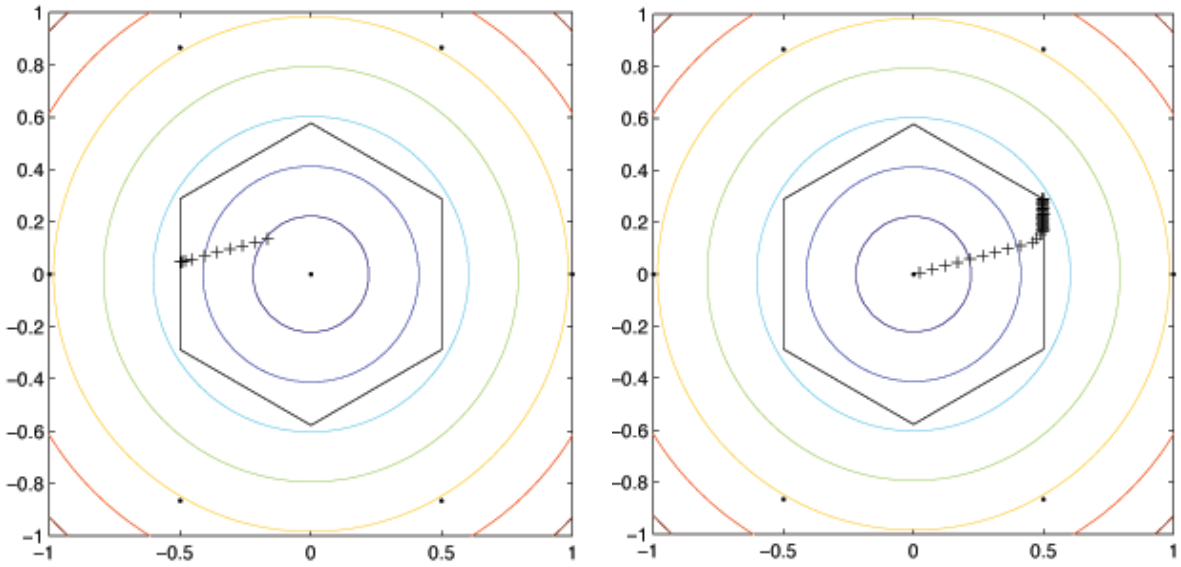

Fig. 13: Locating the deep holes of the A_2 lattice utilizing Successive Polling (left) and Λ-MADS (right). The hard constraints are indicated in black; the cost funtion contours are plotted as well. The inability of the SP algorithm to handle constraints prevents convergence; Λ-MADS maintains convergence and locates the deep hole.

XII. A Numerical Example: Locating the Deep Hole of a Lattice

An example of a relevant optimization problem where a pattern search fails due to constraint behavior was realized by the authors while performing numerical analysis of efficient lattices (see Bewley, Belitz, & Cessna (2011)). The challenge is to calculate the location of a deep hole belonging to the origin node of a given 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 the point lies within the voronoi cell (that is, if the point is closer to the origin than any other lattice point in the cloud). Then, the objective is to locate the point furthest from the origin that remains in the voronoi cell.

Pictorially, this is quite clear in Figure 8, where the cost function contours are plotted for the A_2 lattice. In the interest of remaining computationally feasible, the constraints can only be implemented as hard constraints. 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. This presents an interesting research problem where traditional derivative-based algorithms cannot be applied and SP and other simple GPS algorithm fail to converge.

The voronoi cell boundaries are treated as hard constraints. Under this problem definition, as can be seen in Figure 8, the Successive Polling algorithm fails to converge to a deep hole. 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, & Cessna, (2011).

XIII. Extension

MADS and other GPS algorithms are frequently extended under more generalized optimization frameworks. The Poll step of a MADS or GPS algorithm can be combined with a globally-convergent surrogate-based Search algorithm, as was originally proposed and implemented in the LABDOGS algorithm introduced by Belitz & Bewley, 2011. Any GPS iteration can be implemented in a given SMF-type framework. Thus, it an era of sophisticated numerical optimization of physical systems, it is important to optimize the performance of each component of modern optimization algorithms.

Within an LABDOGS-type framework, the philosophy of minimizing the size of the Poll set to increase efficiency of the overall optimization algorithm is well documented. SMF orginally utilized a minimal positive basis SP Poll step; later iterations substituted LTMADS, then OrthoMADS, though in the latter the possibility of the detrimental effect on efficiency of a maximal positive Poll set was introduced (see Yang & Marsden, 2011). In the literature, a bias towards a minimal positive basis has been evident for some time. As LTMADS indicates, finding a good minimal positive basis on the Cartesian lattice

is challenging and often sub-optimal. The Λ-MADS algorithm is compatible with the lattices used in LABDOGS, allowing a hybridization of an efficient MADS step with the globally convergent Search algorithms implemented in LABDOGS.

XIV. Conclusions

The work detailed herein introduces efficient lattices to quantize parameter space and coordinate a new Mesh Adaptive Direct Search method called Λ-MADS. Λ-MADS utilizes any of a number of highly efficient lattices as an underlying grid. The algorithm is an analogue to the OrthoMADS algorithm, and maintains all the convergence characteristics of OrthoMADS while improving upon the convergence rate by a factor of approximately three.

The advantages of Λ-MADS over LTMADS are twofold: the poll set of Λ-MADS is (asymptotically) perfectly uniform, whereas LTMADS by construction locates poll sets that vary both angularly and radially. Also, Λ-MADS reuses poll orientations, allowing the algorithm to converge in the presence of hard constraints.

The advantages of Λ-MADS over OrthoMADs are threefold: the poll set of Λ-MADS is uniform from poll set to poll set, whereas OrthoMADS has radial nonuniformity due to the Cartesian grid. OrthoMADS uses a Halton Sequence to generate the polling directions; numerical tests indicate that a poll direction that becomes dense in the hypersphere rather than hypercube offers a performance increase of approximately 10%; thus, Λ-MADS uses a poll direction that is dense in the hypersphere. Finally, Λ-MADS is able to locate highly uniform minimal positive basis where OrthoMADS can only locate uniform maximal basis. The performance penalty incurred by utilizing a maximal rather than a minimal basis is approximately a factor of three in $n = 8$, and the performance difference becomes greater as the dimension of the problem increases.

In statistically relevant numerical analysis, Λ-MADS outperforms OrthoMADS by up to a factor of three in number of function evaluations necessary to reach convergence on quadratic cost functions. On the *n*-dimensional Rosenbrock test function the performance difference is even greater. Further numerical tests implicate the maximal positive basis of OrthoMADS as the primary performance limitation of the algorithm.

The numerical results presented herein are clear: Λ-MADS is a highly competitive MADS optimization algorithm for modern, expensive numerical optimization.

References

[1]

[2] 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.
Abramson, MA, Audet, C, Dennis, Jr, JE (2005) Nonlinear Programming by Mesh Adaptive Direct Searches *Report TR05-13*, Rice University, Texas, 2005

[3] Audet, C, & Dennis, Jr, JE (2006) Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, **17** 188217.

[4] Baake, M, & Grimm, U (1997) Coordination sequences for root lattices and related graphs *Zeitschrift für Kristallographie* **212**, 253-256.

[5] 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**, 113.

[6] Conway, JH, & Sloane, NJA (1997) Low-dimensional lattices VII. Coordination sequences *Proc. R. Soc. Lond* A **453**, 2369-2389.

[7] Conway, JH, & Sloane, NJA (1998) *Sphere Packings, Lattices, and Groups*, Springer.

[8] Coope, ID, & Price, CJ (2001) On the convergence of grid-based methods for unconstrained optimization. *SIAM J. Optim.*, **11**, 859869.

[9] 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), 315329. SIAM.

[10] Grünbaum, B (2002) *Convex polytopes*, second edition. Springer.

[11] 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**, 577582, Chicago.

[12] Jones, DR (2001) A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization* **21**, 345-383.

[13] 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**, 97106.

[14] Mockus, J (1994) Application of Bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, **4**, 347365.

[15] 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.*

[16] Stuckman, BE (1988) A global search method for optimizing nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics*, **18**, 965977.

[17] Torczon, V (1997) On the convergence of pattern search algorithms. *SIAM J. Optim.*, **7**, 1-25.

[18] Torn, A, & Zilinskas, A (1987) *Global Optimization*, Springer.