Fast construction of efficient cut cell quadratures

Samuel F. Potter*

July 31, 2025

Abstract

We survey existing optimization-based algorithms for computing efficient cut cell quadratures in two and three dimensions. We consider the convex geometry of the optimization problems being solved and conditions under which positive quadratures are obtained. Focusing on level set domains sampled with a uniform background grid, we evaluate in turn algorithms based on least squares, Steinitz elimination, nonnegative least squares, basis pursuit, linear programming, and Gauss-Newton iteration—these methods form a quadrature toolbox, which can be combined in different ways. We find three particularly effective methods for generating rules. First, we use Xiao and Gimbutas' method for node elimination to eliminate nodes from nonnegative least squares quadratures. Second, we find that it is possible to extract a warm start for Gauss-Newton iteration from the first primal iterate of the primal-dual iteration used to solve Ryu and Boyd's linear program. Third, we find that the first dual variable can be immediately used to produce a warm start for Gauss-Newton. By interpreting this dual variable as the vector rejection of the Ryu-Boyd residual function with respect to the polynomial space being integrated, we see that in 1D this is the same as initializing Gauss-Newton with the local minima of a Legendre polynomial which, in turn, are asymptotic to the Gauss-Legendre abscissae for large degree. Numerical results are presented.

1 Introduction

In 1D, Gaussian quadrature is the method of choice for integrating general, smooth functions. In higher dimensions, a variety of Gaussian—or Gauss-like (i.e., "efficient")—rules have been developed, typically for canonical domains (squares, disks, triangles, etc.). For more complicated domains, some combination of mapping and decomposition (i.e., meshing) are used to combine simple, canonical quadrature rules into composite ones. On the other hand, certain simulation methods like the cut finite element method [5] and other immersed methods superimpose regular background meshes on top of unstructured domains, computing moments over unstructured cut cells. As it has long been recognized that finite element assembly is best accomplish through the use of quadrature rules, there is a recognized need for efficient quadrature rules on cut cells. Beyond this need, the ability to quickly and accurately integrate arbitrary smooth or analytic functions on unstructured geometry is of manifestly evident utility. This work collects results from the literature on using numerical optimization to compute such quadrature rules, presents several new fast algorithms, and outlines some directions for further research.

1.1 Related work

We give a high-level overview of existing successful algorithms for quadrature generation which use some form of numerical optimization or numerical linear algebra.

*sam@mcneel.com — Robert McNeel & Associates

Huybrechs' accurate equispaced quadratures. Early work on using numerical methods to design special quadratures was done by Huybrechs [15]. In this work, it was shown that stable quadratures with equispaced nodes (or a subset thereof) can be designed for the interval $[a, b] \subseteq \mathbb{R}$ by solving least squares and nonnegative least squares problems. Let $x_1 < \cdots < x_n$ be an equispaced grid such that $x_1 = a$ and $x_n = b$ and let $\mathbb{P}_m$ being the polynomial space being integrated. Huybrechs shows that there exists some $N = O(m^2)$ such that for all $n \geqslant N$, the least squares quadrature (see Section 3.2) has positive weights, providing a fast algorithm for computing these weights which exploits properties of polynomials which are orthogonal with respect to the discrete uniform measure supported on the equispaced grid. He also presents some preliminary results showing that the nonnegative least squares quadrature (see Section 3.4) computed on a grid with $O(m^2)$ nodes produces a positive quadrature with $m + 1$ nonzero weights, where the space being integrated is $\mathbb{P}_m$. Huybrechs uses MATLAB's implementation of nonnegative least squares (`lsqnonneg`), which uses Lawson and Hanson's active set method [18] but does not explore the algorithmic details further.

Vioreanu-Rokhlin quadrature. If $\Omega \subseteq \mathbb{C}$ is compact and convex and V is a finite-dimensional subspace of $L^2(\Omega)$, Vioreanu and Rokhlin showed that the eigenvalues of the complex multiplication operator $\mathcal{M}_{x+iy} : V \to L^2(\Omega)$ lie within Ω, providing numerical evidence that these eigenvalues are a reasonable warm start for a nonlinear rootfinder applied to the moment-matching equations [25]. If Ω possesses any symmetries, the eigenvalues of the multiplication operator have the same symmetries. After the eigenvalues are computed, quadrature weights are computed using least squares, and the subsequent quadrature is optimized by applying Gauss-Newton to the moment-matching equations (any symmetries that might exist in the eigenvalues are not enforced). Unfortunately, if Ω is nonconvex, the eigenvalues may not lie within Ω, somewhat limiting the applicability of the method to canonical two-dimensional domains which are convex or nearly convex. The method also has trouble generating quadratures for domains with continuous symmetry groups. We note that the connection between quadrature and the multiplication operator, along with extensions of these ideas to higher dimenions, is explained in the work of Collowald and Hubert [7].

Ryu-Boyd quadrature. Ryu and Boyd explored using linear programming to design positive quadratures [22]. They formulate the problem as an infinite-dimensional linear program (LP) over a measure space, and produce a finite-dimensional LP by approximating the continuous measures with discrete measures. They prove that the infinite-dimensional LP recovers Gaussian quadratures on the interval in one dimension. Practically speaking, they lay down a regular grid of quadrature nodes inside Ω and solve an LP whose primal variables are the quadrature weights at each node. The LP has nonnegativity constraints enforcing positivity and equality constraints ensuring that the polynomial space is integrated. The choice of cost function for the LP is open-ended and has a dramatic effect on the type of quadrature produced—they illustrate the effect of different choices with numerical examples. Once the LP is solved, the quadrature must be extracted and further optimized. The strength of the method is its adaptability to different integration domains. They do not consider algorithmic details or carry out performance studies—we consider this method in more detail in Section 3.7 and show how it can be made efficient in Section 4.

Xiao-Gimbutas quadrature. The goal of the work by Xiao and Gimbutas [27] is to develop an algorithm for computing moment-matched (see Section 2.3) positive quadratures on canonical domains which have nearly the minimum number of nodes possible. In summary, they start by creating a quadrature for the domain through some combination of mapping and decomposition, where known positive quadratures are used for each of the resulting subdomains. Optionally, they explicitly enforce any symmetries of the domain that are desired in the quadrature, reducing the number of variables in the moment-matching equations. To create a warm start for a Gauss-Newton iteration, they select a minimal subset of the nodes of the mapped and decomposed quadrature using a rank-revealing QR decomposition, along the lines of what is suggested in Martinsson et al. [19]. They then proceed to solve the moment-matching equations and eliminate the "least significant" node (see Section 3.8) one at a time, using recursive backtracking to search over all sequences of eliminated nodes until a maximally efficient quadrature is found. In this way, they are able to compute very nearly optimal quadratures, although the cost of their algorithm presumably is quite high (they do

not report timing statistics). The goal of their work is very different: they seek nearly optimal quadratures which are usable on standard mesh elements (e.g. triangles, tetrahedra, quadrilaterals, etc.), and are content with paying a high upfront cost.

Other quadratures based on numerical optimization. Thiagarajan and Shapiro solve the moment matching equations with a fixed grid using a QR decomposition [24]. Saye gave an algorithm for recursively constructing mapped tensor product Gauss rules on cut cells defined as the region bounded by the graph of a function, specifically for use in a discontinuous Galerkin simulation [23]. Glaubitz considers basis pursuit and least squares quadratures for use with experimental data, typically high-dimensional; some results on the existence and positivity of these quadratures are proved [11, 12, 13]. Other similar studies on generating high-dimensional quadratures using methods from optimization have been carried out [17, 16]. With monte carlo applications in mind, Elefante solves a sequence of nonnegative least squares problems on low-discrepancy point sets until a small enough error in the moment-matching equations is achieved [9]. Bui et al. explore using moment-matched cut cell quadratures for elastoplastic simulations using the finite cell method [4]. Garhuom et al. uses nonnegative least squares to prune adaptive quadtree rules [10].

2 Preliminaries

2.1 Notation and basic definitions

Throughout, we let $d > 0$ denote the dimension of the ambient space $\mathbb{R}^d$. A multi-index is written $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$, where $\alpha_i \geqslant 0$ for each $i = 1, \ldots, d$. If $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, a d-variate monomial is written:

$$\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdots x_d^{\alpha_d}. \tag{1}$$

We write $|\boldsymbol{\alpha}| = \alpha_1 + \cdots + \alpha_d \geqslant 0$ for the degree of a multi-index. Let I be an index set and for each $i \in I$, let $\boldsymbol{\alpha}_i$ be a multi-index and let $c_i \in \mathbb{R}$ be a coefficient. The *total degree* of the polynomial $p(\boldsymbol{x}) = \sum_{i \in I} c_i \boldsymbol{x}^{\boldsymbol{\alpha}_i}$ is:

$$\deg p = \max_{i \in I} |\boldsymbol{\alpha}_i|. \tag{2}$$

We note that the *partial degree* of p is defined to be $\max_i \max_j (\boldsymbol{\alpha}_i)_j$, which coincides with spaces of tensor product polynomials. We will restrict our attention to spaces of polynomials up to a given total degree on account of their importance in approximation. The set of d-variate polynomials of total degree at most N is denoted:

$$\mathbb{P}_N^d = \operatorname{span} \{ \boldsymbol{x}^{\boldsymbol{\alpha}} : |\boldsymbol{\alpha}| \leqslant N \}, \tag{3}$$

where $\dim \mathbb{P}_N^d = \binom{N+d}{d}$. Note that the dimensions of $\mathbb{P}_N^2$ and $\mathbb{P}_N^3$ are the triangular and tetrahedral numbers. Throughout, we let $m = \dim \mathbb{P}_N^d$.

2.2 Background on quadrature

Let $n > 0$ be the *order* of a quadrature rule with weights $w_1, \ldots, w_n \in \mathbb{R} \setminus \{0\}$ and nodes $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$. Define the operator:

$$\mathcal{Q}[f] = \sum_{i=1}^{n} w_i f(\boldsymbol{x}_i), \tag{4}$$

where f is a scalar-valued function defined on $\mathbb{R}^d$. Let $\Omega \subseteq \mathbb{R}^d$ and let $\mu : \Omega \to (0, \infty)$ be a density. If $\mathcal{Q}[f]$ is a good approximation of $\int_\Omega f(\boldsymbol{x}) \mu(\boldsymbol{x}) d\boldsymbol{x}$, then we call $\mathcal{Q}$ a *quadrature* with respect to the set Ω and the density μ. The number of *degrees of freedom* of the quadrature is $\operatorname{dof}(\mathcal{Q}) = (d+1)n$. Although the methods presented here can be easily adapted to a nonuniform density (or more general measure) μ, we restrict our attention to $\mu \equiv 1$ for simplicity and because of its centrality in applications.

For $d = 2, 3$, we refer to a subset $\Omega \subseteq [-1, 1]^d$ such that $\partial\Omega$ and $\partial[-1, 1]^d$ overlap nontrivially as a *cut cell*. The $\boldsymbol{\alpha}$*th monomial moment* on the cut cell Ω is:

$$I_{\boldsymbol{\alpha}} = \int_{\Omega} \boldsymbol{x}^{\boldsymbol{\alpha}} d\boldsymbol{x}, \tag{5}$$

where $\boldsymbol{x}^{\boldsymbol{\alpha}} = \boldsymbol{x}_1^{\alpha_1} \cdots \boldsymbol{x}_d^{\alpha_d}$. If $\varphi_1, \ldots, \varphi_m$ form a basis for $\mathbb{P}_N^d$, we define moments in terms of each φ_i analogously. E.g., the φ_i moment is the integral $\int_\Omega \varphi_i(\boldsymbol{x}) d\boldsymbol{x}$. If $\mathcal{Q}[p] = \int_\Omega p(\boldsymbol{x}) d\boldsymbol{x}$ for all $p \in \mathbb{P}_N^d$, then $\mathcal{Q}$ *integrates* $\mathbb{P}_N^d$. Following Xiao and Gimbutas [27], the *efficiency* of an order n quadrature that integrates $\mathbb{P}_N^d$ is:

$$\operatorname{eff}(\mathcal{Q}) = \frac{\dim \mathbb{P}_N^d}{\operatorname{dof}(\mathcal{Q})} = \frac{m}{(d+1)n}. \tag{6}$$

Our focus is developing fast algorithms for quickly computing reasonably efficient quadratures. We expect $0 \leqslant \operatorname{eff}(\mathcal{Q}) \leqslant 1$ to hold, which is indeed the case, although the upper bound is a theorem. Gaussian quadrature in 1D obtains $\operatorname{eff}(\mathcal{Q})$, but little about the maximum efficiency of multidimensional quadratures is known.

Let $n > 0$, let $x_1, \ldots, x_n$ be the zeros of the nth Legendre polynomial, and let $w_i = \int_{-1}^{1} \ell_i(x) dx$, where $\ell_i(x)$ is the ith Lagrange basis polynomial determined by the nodes $x_1, \ldots, x_n$. The corresponding quadrature $\mathcal{Q}_1$ is the order n Gauss-Legendre rule. This quadrature integrates all polynomials in $\mathbb{P}_{2n-1}^1$, hence $\operatorname{eff}(\mathcal{Q}_1) = 1$. Quadratures for rectangles in dimensions greater than one can be obtained by taking tensor products of Gauss-Legendre rules.

Example 2.1 (efficiency of tensor product Gauss-Legendre quadrature)**.** Let $d > 1$ and define the *order n tensor-product Gauss-Legendre quadrature* $\mathcal{Q}_d$ by the formula:

$$\mathcal{Q}_d[f] = \sum_{i_1=1}^{n} \cdots \sum_{i_d=1}^{n} w_{i_1} \cdots w_{i_d} f(x_{i_1}, \ldots, x_{i_d}). \tag{7}$$

This quadrature integrates all monomials $\boldsymbol{x}^{\boldsymbol{\alpha}}$ where $\max_i \alpha_i \leqslant 2n-1$; hence, it integrates all monomials $\boldsymbol{x}^{\boldsymbol{\alpha}}$ for which $|\boldsymbol{\alpha}| \leqslant 2n-1$. To reiterate: the former set of monomials is the set of monomials of partial degree at most $2n-1$, the latter set is $\mathbb{P}_{2n-1}^d$. The latter is a subset of the former, and, over all possible N, the largest $\mathbb{P}_N^d$ so contained. Consequently, with respect to the total degree, we can see that $\mathcal{Q}_d$ "over-integrates".

Since $\operatorname{dof}(\mathcal{Q}_d) = (2n)^d$, and since $\dim(\mathbb{P}_{2n-1}^d) = \binom{2n-1+d}{d}$, we have:

$$\operatorname{eff}(\mathcal{Q}_d) = \frac{\binom{2n-1+d}{d}}{(d+1)n^d} = \frac{2n(2n+1)\cdots(2n+d-1)}{(d+1)d!n^d} = \frac{(2n)^d + \cdots}{(d+1)!n^d} \sim \frac{2^d}{(d+1)!} \text{ as } n \to \infty. \tag{8}$$

This recovers $\operatorname{eff}(\mathcal{Q}_1) = 1$, and $\operatorname{eff}(\mathcal{Q}_2) \sim 2/3$ and $\operatorname{eff}(\mathcal{Q}_3) \sim 1/3$. Clearly, $\operatorname{eff}(\mathcal{Q}_d) \sim 0$ if $d \to \infty$ and $n \to \infty$ simultaneously.

We generically refer to a quadrature for a known, simple geometry (a *canonical domain*) with an efficiency higher than what could otherwise be achieved by "conventional means" as an *efficient quadrature*. For $d > 1$, very little is known about the maximum achievable efficiency of quadratures for canonical domains, let alone more exotic ones. If Ω isn't canonical, a standard procedure for numerical integration is to possibly decompose Ω into a disjoint union of subdomains, and define each subdomain as the image of a canonical domain. By using a known quadrature for the preimage of each subdomain and suitably adjusting its weights we get a *mapped quadrature*. Even if decomposition can be avoided, mapping can have a deleterious effect on efficiency.

Example 2.2 (Efficiency of mapped quadratures.)**.** Let Ω be a deformation of the unit square with one side given by the polynomial $h(x)$:

$$\Omega = \left\{(x, y) \in \mathbb{R}^2 : 0 \leqslant x \leqslant 1 \text{ and } 0 \leqslant y \leqslant h(x)\right\}. \tag{9}$$

Integrating the monomial $x^p y^q$ over Ω gives:

$$\int_\Omega x^p y^q dA(x,y) = \int_0^1 \int_0^1 x^p (yh(x))^q y dy dy =: \int_0^1 \int_0^1 F(x,y). \tag{10}$$

The polynomial degree with respect to x of F is $r(p,q) = p+(q+1)\deg h$ while the degree with respect to y is q. Consequently, to integrate $x^p y^q$ we can first apply integration by substitution as above and then integrate over the unit square using a Gauss-Legendre rule. This requires an $\lceil r(p,q)/2 \rceil$-point rule to integrate over x and a $\lceil q/2 \rceil$-point rule for y; consequently, to integrate each monomial in $\mathbb{P}_n^2$, we need a rule with $\lceil n/2 \rceil$ points to integrate in y and $\lceil r^*/2 \rceil$ points to integrate in x, where:

$$r^* = \max_{0 \leqslant p \leqslant n} r(p, n-p). \tag{11}$$

Denote this rule by $\mathcal{Q}_{n,h}$. Its efficiency satisfies:

$$\operatorname{eff}(\mathcal{Q}_{n,h}) \sim \left(\frac{n(n+1)}{2}\right) \Big/ \left(\frac{3}{4} r^* n\right) = \frac{2}{3} \cdot \frac{n+1}{\max_{0 \leqslant p \leqslant n} \Big(p + (n-p+1)\deg h\Big)}. \tag{12}$$

Since the maximum in the denominator is bounded below by $(n+1)\deg h$, we have:

$$\operatorname{eff}(\mathcal{Q}_{n,h}) \lesssim \frac{2}{3} \Big(\deg h\Big)^{-1}. \tag{13}$$

This recovers $\operatorname{eff}(\mathcal{Q}_2) \sim 2/3$ if h is linear, as expected. But we can now see that if h is of even moderate degree, the efficiency of the rule decays rapidly, resulting in a large number of quadrature points needed to integrate $\mathbb{P}_n^2$ exactly.

Of course, we may still do numerical integration accurately enough for some purpose without integrating the desired polynomial space exactly, and such inexact rules are worthy of consideration (engineering is replete with such hacks). In this work we eliminate this consideration by endeavoring to integrate the target space close to machine precision.

The need for native cut cell quadratures. The particular focus of this work is designing quadrature rules for cut cells, primarily to be used for immersed methods. In this case, an arbitrary segment of the boundary is restricted to a grid cell, with that subset of the domain to be used as an integration domain. Basic approximation theory tells us that if the continuity class of the boundary is low, mapped quadrature is doomed to failure. For example, if $h(x)$ is a polynomial approximating the function $y(x) = |x - 1/2| + 1/2$ in Example 2.2, it is well known that tens of thousands (if not hundreds of thousands) of coefficients (e.g., computed in the Chebyshev basis) are needed for a decent approximation. The obvious resolution to this problem is to further subdivide Ω—that is, to build a mesh on it. But avoiding meshing is the motivation for choosing an immersed method in the first place. By punting and moving the possibility of mesh failure elsewhere, we have not solved our original problem, and the promise of the method is unrealized. Consequently, there is a need to construct *native* cut cell quadrature rules without recourse to meshing. It turns out that not only can native quadrature rules be constructed without meshing, they are dramatically more efficient than the alternative. In particular, we will find that it is straightforward to construct rules that integrate $\mathbb{P}_n^d$ with an efficiency of $1/(d+1)$. With a little work, we can push that efficiency even higher.

2.3 Moment-matched quadratures

Quadratures for non-canonical domains can be generated using numerical optimization to ensure that each moment is correctly integrated. These *moment-matched quadratures* are our focus. The approach is quite general and can be used to design quadratures for a wide variety of function spaces. For simplicity, and since it is the case of the greatest practical importance, we restrict our attention to $\mathbb{P}_N^d$.

Definition 2.1. The *moment-matching equations* for the quadrature $\mathcal{Q}$ with weights $w_1, \ldots, w_n$ and nodes $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ on Ω with respect to $\mathbb{P}_N^d = \operatorname{span}\{\varphi_1, \ldots, \varphi_m\}$ are:

$$I_i = \mathcal{Q}[\varphi_i] = \sum_{j=1}^{n} w_j \varphi_i(\boldsymbol{x}_j), \qquad i = 1, \ldots, m. \tag{14}$$

Letting $\boldsymbol{w} = (w_1, \ldots, w_n)$ and $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{R}^d)^n$, the residual function for the moment-matching equations is $\boldsymbol{F} : \mathbb{R}^n \times \mathbb{R}^{dn} \to \mathbb{R}^m$ where $\boldsymbol{F}_i(\boldsymbol{w}, \boldsymbol{X}) = I_i - \mathcal{Q}[\varphi_i]$.

A quadrature integrates $\mathbb{P}_N^d$ if the moment-matching equations are satisfied. If we fix $\boldsymbol{X}$ and find $\boldsymbol{w}$ that satisfies (14), then the moment-matching equations are linear in $\boldsymbol{w}$. Defining the matrix

$$\boldsymbol{V} = \begin{bmatrix} \varphi_1(\boldsymbol{x}_1) & \cdots & \varphi_m(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \varphi_1(\boldsymbol{x}_n) & \cdots & \varphi_m(\boldsymbol{x}_n) \end{bmatrix} \in \mathbb{R}^{n \times m}, \tag{15}$$

this linear system reads

$$\boldsymbol{V}^\top \boldsymbol{w} = \boldsymbol{I}. \tag{16}$$

If we seek a pair $(\boldsymbol{w}, \boldsymbol{X})$ that simultaneously satisfies (14), then the system is nonlinear, and the computation of the Jacobian of the residual function becomes important. It reads:

$$\mathsf{D}\boldsymbol{F} = \begin{bmatrix} \mathsf{D}_{\boldsymbol{w}}\boldsymbol{F} & \mathsf{D}_{\boldsymbol{X}}\boldsymbol{F} \end{bmatrix} = \begin{bmatrix} \varphi_1(\boldsymbol{x}_1) & \cdots & \varphi_1(\boldsymbol{x}_n) & w_1\mathsf{D}\varphi_1(\boldsymbol{x}_1) & \cdots & w_n\mathsf{D}\varphi_1(\boldsymbol{x}_n) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \varphi_m(\boldsymbol{x}_1) & \cdots & \varphi_m(\boldsymbol{x}_n) & w_1\mathsf{D}\varphi_m(\boldsymbol{x}_1) & \cdots & w_n\mathsf{D}\varphi_m(\boldsymbol{x}_n) \end{bmatrix}, \tag{17}$$

where $\mathsf{D}\varphi_i = \begin{bmatrix} \partial\varphi_i/\partial x_1 & \cdots & \partial\varphi_i/\partial x_d \end{bmatrix} \in \mathbb{R}^{1 \times d}$. There are a variety of approaches that have been taken to solve the moment-matching equations. Our work combines and builds on several of them, which we recapitulate in the following sections.

2.4 Existence of positive quadratures

Although the task of finding a stable, positive quadrature appears daunting at first sight, Martinsson et al. [19] showed that a stable but not necessarily positive quadrature exists for a set of bounded functions where

1) the integration region is taken to be essentially any set,
2) the quadrature has the same number of nodes as the number of functions being integrated,
3) and the quadrature can be computed from a suitable discretization of the set using a rank-revealing QR decomposition [14].

Such a quadrature is often a reasonable choice. Nevertheless, we seek a quadrature with positive weights since they are expected by practitioners and are (arguably) more stable.

Theorem 1. *Let $\Omega \subseteq \mathbb{R}^d$ be a compact set, let $m > 0$, let $\varphi_1, \ldots, \varphi_m : \Omega \to \mathbb{R}$ be a set of m linearly independent functions, and let $\mathcal{P} = \operatorname{span}\{\varphi_1, \ldots, \varphi_m\}$. Then, there exists a positive quadrature with order not greater than m, with nodes in Ω, and which integrates $\mathcal{P}$.*

Proof. This is a specialization of Tchakaloff's theorem to the present setting [**?**]. Davis provided a constructive proof of Tchakaloff's theorem [8]. We provide a similar constructive proof adapted to this setting in Sections 3.2 and 3.3. □

In the context of this paper, we will refer to Theorem 1 as Tchakaloff's theorem, bearing in mind that the actual theorem of Tchakaloff is more general.

In the foregoing discussion, we consider quadratures which are positive, minimal (satisfy Tchakaloff's theorem), and *exact*—i.e., which exactly satisfy the moment-matching equations for the quadrature. If we relax the last requirement, the existence proof is simpler.

Theorem 2. *Let $\boldsymbol{X} \subseteq \mathbb{R}^d$ be point set and let $n = |\boldsymbol{X}| < \infty$. Let $\mathcal{P} = \operatorname{span}\{\varphi_1, \ldots, \varphi_m\}$. Let $\boldsymbol{V} = \begin{bmatrix}\varphi_1(\boldsymbol{X}) & \cdots & \varphi_m(\boldsymbol{X})\end{bmatrix}$ and let $\boldsymbol{I} = (\int_\Omega \varphi_1(\boldsymbol{x})d\boldsymbol{x}, \ldots, \int_\Omega \varphi_m(\boldsymbol{x})d\boldsymbol{x})$. Assume that $\boldsymbol{w}^*$ is an optimum of the minimization problem:*

$$\begin{aligned} &\textit{minimize} && \left\|\boldsymbol{V}^\top \boldsymbol{w} - \boldsymbol{I}\right\|_2 \\ &\textit{subject to} && \boldsymbol{w} \geqslant \boldsymbol{0}. \end{aligned} \tag{18}$$

Then, there exists $\boldsymbol{w}^\star \geqslant \boldsymbol{0}$ such that $\|\boldsymbol{V}^\top \boldsymbol{w}^\star - \boldsymbol{I}\|_2 \leqslant \|\boldsymbol{V}^\top \boldsymbol{w}^* - \boldsymbol{I}\|_2$ and $\operatorname{card}(\boldsymbol{w}^\star) \leqslant d$.

Proof. First, we recall Caratheodory's theorem, which states that any point in the convex hull of a set in $\mathbb{R}^d$ can be written as a convex combination of $d+1$ points in the original set [2]; equivalently, any point in the conic hull of a set in $\mathbb{R}^d$ be written as the conic combination of d points taken from the generating set.

Now, let $\boldsymbol{R}^* = \boldsymbol{I} - \boldsymbol{V}^\top \boldsymbol{w}^*$. Of course, $\boldsymbol{R}^* \neq \boldsymbol{0}$ in general. Since $\boldsymbol{V}^\top \boldsymbol{w}^* = \boldsymbol{I} - \boldsymbol{R}^*$, we can see that $\boldsymbol{I} - \boldsymbol{R}^*$ is a conic combination of the columns of $\boldsymbol{V}^\top$; that is, $\boldsymbol{I} - \boldsymbol{R}^* \in \operatorname{cone}\{\boldsymbol{\varphi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\varphi}(\boldsymbol{x}_n)\}$. Then, by Caratheodory's theorem, $\boldsymbol{I} - \boldsymbol{R}^*$ can be written as a conic combination of at most d points in $\{\boldsymbol{\varphi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\varphi}(\boldsymbol{x}_n)\}$. Let $\boldsymbol{w}^\star \in \mathbb{R}^n$ be the nonnegative vector whose jth component is given by Caratheodory's theorem and zero otherwise; consequently, $\boldsymbol{V}^\top \boldsymbol{w}^\star = \boldsymbol{I} - \boldsymbol{R}^*$. □

Theorem 2 is *not* constructive. Caratheodory's theorem merely indicates that an m-sparse combination of columns of $\boldsymbol{V}^\top$ exists, but not how to compute it.

3 Quadrature toolbox

3.1 Discretizing the cut cell

We can view moment matching as a means of constructing a quadrature by finding a discrete measure which approximates a continuous measure by agreeing exactly on a finite-dimensional function space. Specifically, for the cut cell Ω, our goal is to approximate the uniform measure on Ω with a discrete measure with positive weights and with support contained inside Ω:

$$\mu_n(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \delta(\boldsymbol{x} - \boldsymbol{x}_i), \qquad w_i > 0, \quad \boldsymbol{x}_i \in \Omega, \quad 1 \leqslant i \leqslant n. \tag{19}$$

so that:

$$\int_\Omega \varphi_j(\boldsymbol{x})d\boldsymbol{x} = \int_\Omega \varphi_j(\boldsymbol{x})d\mu_n(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \varphi_j(\boldsymbol{x}_i), \qquad 1 \leqslant j \leqslant m, \tag{20}$$

which is equivalent to the moment-matching equations. It is easier to compute the weights with the nodes fixed rather than simultaneously choosing both weights and nodes. The algorithms presented in this section are unified in the sense that they all start by discretizing Ω into a grid of n nodes where $n \gg m$ and then computing the weights. The first approach computes a dense quadrature where $w_i > 0$ for all i, but the remaining approaches use techniques from sparse optimization to compute a minimal number of nonzero weights ($m \leqslant n$), thereby selecting a subset of the original node set.

We do not spend any time on the question of how best to discretize Ω. In our numerical experiments, we always follow the same procedure:

Algorithm 3.1: Discretize cut cell Ω with a uniform grid

1) Find scalars $a_1, \ldots, a_d$ and $b_1, \ldots, b_d$ such that $a_j < b_j$ for each j, such that $\Omega \subseteq [a_1, b_1] \times \cdots \times [a_d, b_d]$, and such that the a_j's are maximized and the b_j's are minimized. (*Find the smallest bounding box containing Ω.*)
2) Map $[a_1, b_1] \times [a_d, b_d]$ to $[-1, 1]^d$, denote the mapping by $\boldsymbol{F}$.
3) Discretize $[-1, 1]^d$ into a uniform regular grid with spacing $h > 0$.

4) Let $\boldsymbol{X}$ be the set of grid nodes contained in $\boldsymbol{F}(\Omega)$.

Clearly, following this discretization, we must use the change of variables formula

$$\int_\Omega \varphi(\boldsymbol{x})d\boldsymbol{x} = \int_{\boldsymbol{F}^{-1}(\Omega)} \varphi(\boldsymbol{F}(\boldsymbol{u}))|\mathsf{D}\boldsymbol{F}(\boldsymbol{u})|d\boldsymbol{u} = \frac{1}{2^d}\prod_{j=1}^{d}(b_j - a_j)\int_{\boldsymbol{F}^{-1}(\Omega)} \varphi(\boldsymbol{F}(\boldsymbol{u}))d\boldsymbol{u}. \tag{21}$$

when computing moments. Rescaling Ω serves two purposes. Most obviously, it allows a more even distribution of grid nodes over Ω which avoids issues with tiny sliver cut cells. However, it also improves the tendency of our choice of residual functions to create Lobatto rules when computing Ryu-Boyd quadratures, as in Section 4.

Huybrechs makes it clear that a simple uniform discretization is a perfectly reasonable choice on a one-dimensional interval [15]. Experimentally, how well this works appears to be independent of the cut cell geometry, provided that the grid is fine enough. Martinsson et al. give strong evidence that this should indeed be the case [19]. To choose a fine enough grid, we again follow Huybrechs, who determines experimentally that the number of nodes needed to integrate $\mathbb{P}_m$ is $\Omega(m^2)$ with a small constant. To this end, to integrate $\mathbb{P}_N^d$, we choose $h = O(N^{-2})$. This results in a grid with $\Omega(N^{2d})$ nodes, albeit with a small constant. This constant can be adapted to individual cut cells by scaling by $\mathrm{vol}(\Omega)$, which can be computed using the algorithms in **??** (indeed, for the monomial and Chebyshev bases, we will compute $\mathrm{vol}(\Omega)$ as a matter of course when we set up the moment-matching equations).

It is intuitively obvious that $h = O(n^{-2})$ is necessary for a uniform grid. We give a heuristic motivation now. The density of quadrature points approaches $\rho(x) = \frac{1}{\sqrt{1-x^2}}$ as $n \to \infty$. As a particular instance, we can consider the Chebyshev nodes $x_j = -\cos(\pi j/n)$ for $j = 0, 1, \ldots, n$. Since the nodes are most closely spaced around the endpoints, we can check what grid spacing $h > 0$ is necessary to ensure, e.g., that x_0 and x_1 can be resolved—that is, the grid is fine enough that not more than one quadrature node is mapped to the same uniform grid point upon being quantized. We have $x_0 = -1$ and $x_1 = -\cos(\pi/n) = -1 + \frac{\pi^2}{2n^2} - O(n^{-4})$. Since $x_1 - x_0 = \frac{\pi^2}{2n^2} - O(n^{-4})$, we can see that it is sufficient to take $h = O(n^{-2})$ to ensure that a quadrature on $[-1, 1]$ with limiting density ρ will be sufficiently resolved.

3.2 Least squares quadrature

As we have seen, both Huybrechs [15] and Glaubitz [11] explored the conditions under which solving rectangular moment-matching equations with fixed nodes produces positive quadratures. With the nodes fixed, the optimization problem to solve is:

$$\text{minimize} \quad \frac{1}{2}\left\|\boldsymbol{V}^\top \boldsymbol{w} - \boldsymbol{I}\right\|_2^2. \tag{22}$$

Since we require $\boldsymbol{V}$ to have $\Theta(m^2)$ rows, this is an underdetermined least squares problem.

The minimum norm solution is easily computed using either the QR or LU decomposition [**?**].

Algorithm 3.2: Compute least squares quadrature

Input: domain $\Omega \subseteq \mathbb{R}^d$, polynomial space $\mathbb{P}$ with $\dim \mathbb{P} = m$, moment vector $\boldsymbol{I}$, discretization of Ω into uniform grid $\boldsymbol{X}$ containing $O(m^2)$ points.
Output: quadrature rule $(\boldsymbol{X}, \boldsymbol{w})$ that integrates $\mathbb{P}$ on Ω.
1) Compute $\boldsymbol{w} = \boldsymbol{V}^\dagger \boldsymbol{I}$ using LU or QR decomposition.

We recall that since (22) is underdetermined, $\boldsymbol{w} = (\boldsymbol{V}^\top)^\dagger \boldsymbol{I}$, where $(\boldsymbol{V}^\top)^\dagger$ denotes the pseudoinverse of $\boldsymbol{V}^\top$. Since $\boldsymbol{V}$ is a wide matrix with linearly independent rows, $(\boldsymbol{V}^\top)^\dagger = \boldsymbol{V}(\boldsymbol{V}^\top\boldsymbol{V})^{-1}$. To use the LU decomposition to compute the minimum norm solution of (22), we LU factorize $\boldsymbol{V}^\top\boldsymbol{V}$. To apply the QR decomposition, we let $\boldsymbol{V} = \boldsymbol{Q}\boldsymbol{R}$ be the reduced QR decomposition of $\boldsymbol{V}$, in which case $\boldsymbol{V}^\dagger = \boldsymbol{Q}\boldsymbol{R}^{-\top}$.

Theorem 3. *Algorithm 3.2 runs in $O(m^{2(d+1)})$ time.*

Proof. This is the case regardless of whether we use the LU or QR decomposition. Since there are $O(m^{2d})$ grid points, $\boldsymbol{V}$ has $O(m^{2d})$ rows and $O(m)$ columns. For the LU decomposition, forming $\boldsymbol{V}^\top \boldsymbol{V}$ requires $O(m^{2(d+1)})$ operations, while subsequently computing the LU factorization of $\boldsymbol{V}^\top \boldsymbol{V}$ requires only $O(m^3)$. Once the LU decomposition of $\boldsymbol{V}^\top \boldsymbol{V}$ is available, computing $(\boldsymbol{V}^\top \boldsymbol{V})^{-1}\boldsymbol{I}$ is $O(m^2)$, and multiplying the result by $\boldsymbol{V}$ is again $O(m^3)$. Hence, the time complexity of computing the least squares quadrature using the LU decomposition is $O(m^{2(d+1)})$ since $d \geqslant 1$. For the QR decomposition, computing the QR decomposition of $\boldsymbol{V}$ using, e.g., modified Gram-Schmidt, takes $O(m^{2(d+1)})$ time. Computing $\boldsymbol{R}^{-\top}\boldsymbol{I}$ is $O(m^2)$ and multiplying the result by $\boldsymbol{Q}$ is $O(m^{2d+1})$. So, solving (22) using a QR decomposition also takes $O(m^{2(d+1)})$ operations. □

The drawback of least squares quadrature is that $\boldsymbol{w}$ will have all nonzero components. That is, the resulting quadrature will be order $O(m^{2d})$. Nevertheless, it occurs later as the initial step when we compute Ryu-Boyd quadratures (Section 4). As we mentioned earlier, Huybrechs devised an asymptotically faster method of constructing least squares quadratures on the interval using properties of discrete orthogonal polynomials. It is unclear whether a similar method could be devised on unstructured domains in two or greater dimensions.

A natural question is whether least squares quadratures are positive, since they approximate the uniform measure supported on Ω. The answer is in the affirmative for a fine enough grid. Huybrechs proved a version of the following theorem in one dimension [15]. The version here is a specialization of a result due to Glaubitz [12].

Theorem 4. *Let $d > 0$ and $N \geqslant 0$. Then, there is an integer $n_0 > 0$ such that for all $n \geqslant n_0$ a set of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ exists such that the resulting least squares quadrature is positive.*

Proof. Let $\Omega \subseteq [-1,1]^d$ be a cut cell, let $n > 0$ be an integer, let $h = 1/n$, and define:

$$\boldsymbol{X}_n = \Omega \cap \left\{ \{-1, -1+h, \ldots, 1\} + \frac{h}{2} \right\}^d. \tag{23}$$

Let $B_{\boldsymbol{x}} = \{\boldsymbol{y} \in \Omega : \|\boldsymbol{x} - \boldsymbol{y}\|_\infty \leqslant h/2\}$ and let $h_{\boldsymbol{x}} = \operatorname{vol}(B_{\boldsymbol{x}})$. Since $\Omega = \bigcup_{\boldsymbol{x} \in \boldsymbol{X}_n} B_{\boldsymbol{x}}$ is disjoint, we have:

$$\operatorname{vol}(\Omega) = \sum_{\boldsymbol{x} \in \boldsymbol{X}_n} \operatorname{vol}(B_{\boldsymbol{x}}) = \sum_{\boldsymbol{x} \in \boldsymbol{X}_n} h_{\boldsymbol{x}}. \tag{24}$$

Next, consider the inner product:

$$(f, g)_{\boldsymbol{X}_n} = \sum_{\boldsymbol{x} \in \boldsymbol{X}_n} h_{\boldsymbol{x}} f(\boldsymbol{x}) g(\boldsymbol{x}). \tag{25}$$

Clearly, if fg is Riemann integrable, then this inner product converges to the $L^2(\Omega)$ inner product as $n \to \infty$. Now, let $\{p_i\}_{i=1}^m$ be a basis for $\mathcal{P}$ such that $p_1 \equiv 1$, and such that $(p_i, p_j)_{\boldsymbol{X}_n} = 0$ if $i \neq j$. Let $\boldsymbol{V} = \begin{bmatrix} p_1(\boldsymbol{X}_n) & \cdots & p_m(\boldsymbol{X}_n) \end{bmatrix}$. Then:

$$\left(\boldsymbol{V}^\top \boldsymbol{V}\right)_{ij} = (p_i, p_j)_{\boldsymbol{X}_n} = \|p_i\|^2_{\boldsymbol{X}_n} \delta_{ij}. \tag{26}$$

Let $\boldsymbol{H} = \operatorname{diag}(h_{\boldsymbol{x}})_{\boldsymbol{x} \in \boldsymbol{X}_n}$ and let $\boldsymbol{D} = \operatorname{diag} \in \mathbb{R}^{m \times m}$ be a diagonal matrix with diagonal entries given by $\boldsymbol{D}_{ii} = \|p_i\|^2_{\boldsymbol{X}_n}$. If we choose $\boldsymbol{w} = \boldsymbol{H}\boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{I}$, then $\boldsymbol{V}^\top \boldsymbol{w} = \boldsymbol{I}$ since $\boldsymbol{V}^\top \boldsymbol{H} \boldsymbol{V} = \boldsymbol{D}$; that is, the moment-matching equations are satisfied. We will show that we can choose n large enough to guarantee $\boldsymbol{w} > 0$. For $i = 1, \ldots, m$, define:

$$c_i = \frac{p_i(\boldsymbol{x})}{\|p_i\|^2_{X_n}} (p_i, p_1)_{L^2(\Omega)}. \tag{27}$$

For a fixed $\boldsymbol{x} \in \boldsymbol{X}_n$, the corresponding quadrature weight can be written:

$$\boldsymbol{w}_{\boldsymbol{x}} = h_{\boldsymbol{x}} \sum_{i=1}^m \frac{p_i(\boldsymbol{x})}{\sum_{\boldsymbol{x}} h_{\boldsymbol{x}} p_i(\boldsymbol{x})^2} \int_\Omega p_i(\boldsymbol{y}) d\boldsymbol{y} = h_{\boldsymbol{x}} \sum_{i=1}^m c_i. \tag{28}$$

For $i = 1$, we have $c_1 = p_1(\boldsymbol{x})\|p_1\|^2_{L^2(\Omega)}/\|p_1\|^2_{X_n} = 1 \cdot \operatorname{vol}(\Omega)/\operatorname{vol}(\Omega) = 1$. Since $(p_i, p_1)_{L^2(\Omega)} \to 0$ as $n \to \infty$, for $i = 2, \ldots, m$ we have $c_i \to 0$. Choose $\epsilon > 0$ satisfying $\epsilon < (m-1)^{-1}$. For $i = 2, \ldots, m$ there exist positive integers $N_2, \ldots, N_m$ such that for $n \geqslant N_i$, we have $|c_i| < \epsilon$. So, let $N \geqslant \max(N_2, \ldots, N_m)$. Consequently:

$$\left| \sum_{i=2}^{m} c_i \right| \leqslant \sum_{i=2}^{m} |c_i| < (m-1)\epsilon < 1, \tag{29}$$

which, for large enough n, gives:

$$\boldsymbol{w}_{\boldsymbol{x}} = h_{\boldsymbol{x}} \cdot \left(1 + \sum_{i=2}^{m} c_i \right) > 0. \tag{30}$$

Now, to ensure that $\boldsymbol{w}_{\boldsymbol{x}}$ for all $\boldsymbol{x} \in \boldsymbol{X}_n$, choose n to be the maximum for each $\boldsymbol{x}$ in the previous step. Crucially, although we can see that $\boldsymbol{w} \to 0$, there exists n beyond which $\boldsymbol{w} \geqslant 0$. □

3.3 Steinitz elimination

Once we have a suboptimal quadrature, we may justifiably ask whether there is any way to incrementally improve the efficiency by eliminating nodes and adjusting weights. If $n > m$, we can do this using linear algebra, eliminating nodes one at a time until $n = m$.

Assume an order $n > m$ positive quadrature exists. Glaubitz shows how to use Steinitz elimination to reduce an order n quadrature to an order m positive quadrature [11].

Theorem 5. *Let $d > 0$, $N > 0$, and $n \geqslant m = \dim \mathbb{P}^d_N$. Let $\mathcal{Q}$ be an order n positive quadrature that integrates $\mathbb{P}^d_N$ on a compact domain Ω. Then, there exists an order m positive quadrature that integrates $\mathbb{P}^d_N$.*

Proof. Let $\boldsymbol{w}$ and $\boldsymbol{X}$ denote the weights and nodes of $\mathcal{Q}$. Let $\varphi_1, \ldots, \varphi_m$ be a basis for $\mathbb{P}^d_N$ and assume that the Vandermonde matrix $\boldsymbol{V} \in \mathbb{R}^{n \times m}$ defined by $\boldsymbol{V}_{ij} = \varphi_j(\boldsymbol{x}_i)$ has full row rank and $\boldsymbol{V}^\top$ has nontrivial nullspace. Let $\boldsymbol{v} \in \mathbb{R}^n$ such that $\boldsymbol{v} \neq 0$ and $\boldsymbol{w} = \boldsymbol{V}^\top \boldsymbol{v}$ has at least one positive component. If we let $k = \arg\max_{1 \leqslant k \leqslant n} \boldsymbol{v}_j/\boldsymbol{w}_j$, then $\boldsymbol{v}_j/\boldsymbol{w}_j > 0$ and $\boldsymbol{w}_i - (\boldsymbol{w}_j/\boldsymbol{v}_j)\boldsymbol{v}_i \geqslant 0$ for each i with equality obtained if $i = j$. Let $\delta\boldsymbol{w} = -(\boldsymbol{w}_j/\boldsymbol{v}_j)\boldsymbol{v}$. Adding $\boldsymbol{V}^\top \delta\boldsymbol{w} = \boldsymbol{0}$ to the moment-matching equations (14) gives $\boldsymbol{V}^\top(\boldsymbol{w} + \delta\boldsymbol{w}) = \boldsymbol{I}$. If we let $\mathcal{I} = (i : 1 \leqslant i \leqslant n \text{ and } \boldsymbol{w}_i + \delta\boldsymbol{w}_i > 0)$, we can see that the quadrature with weights given by $(\boldsymbol{w}_i + \delta\boldsymbol{w}_i : i \in \mathcal{I})$ and nodes given by $(\boldsymbol{x} \in \boldsymbol{X} : i \in \mathcal{I})$ is a positive quadrature with degree at most $n-1$. To compute a positive quadrature of order m, iterate this argument $n - m - 1$ times. □

Clearly, the bulk of the work in Steinitz elimination is in computing the nullspace of $\boldsymbol{V}^\top$, hence the algorithm defined by Theorem 5 has the same complexity. For instance, using Gaussian elimination or a rank-revealing QR decomposition, the complexity is $O(m^2 n) = O(m^{2(d+1)})$, the same as required by Algorithm 3.2.

3.4 Nonnegative least squares quadrature

The theorems in Section 2.4 show that if n is large enough, then an order m positive quadrature exists. Since these proofs are constructive, they could be used as the basis of an algorithm for computing an efficient positive quadrature. However, what this construction suggests is that given a "dense enough" point set, we should be able to select a subset of points which together yield an efficient positive quadrature. In the language of optimization, we could consider the nonnegative least squares problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \left\| \boldsymbol{V}^\top \boldsymbol{w} - \boldsymbol{I} \right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{w} \geqslant 0, \end{aligned} \tag{31}$$

whose global optima constitute the polytope $\{\boldsymbol{w} \in \mathbb{R}^n : \boldsymbol{V}^\top \boldsymbol{w} = \boldsymbol{I} \text{ and } \boldsymbol{w} \geqslant 0\}$. Since there is no unique minimizer, the choice of algorithm used to solve (31) will bias the selection. Again, if n is large enough,

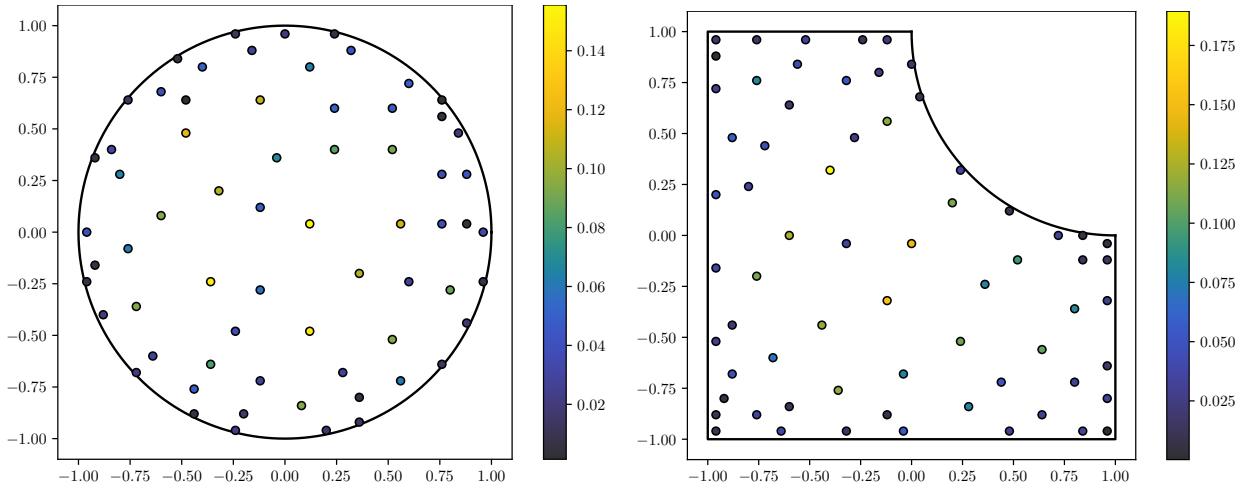

Figure 1: Two simple examples of NNLS quadratures computed using the Lawson-Hanson algorithm, starting from a uniform grid restricted to Ω. *Left*: Ω is the unit disk. *Right*: Ω is the biunit square with a unit disk centered at $(1,1)$ removed. Both quadratures integrate $\mathbb{P}^2_{10}$ (up to numerical roundoff) and have exactly $\dim \mathbb{P}^2_{10} = 66$ points, as expected. The moment-matching equations are solved directly in the monomial basis. In both cases, $\boldsymbol{V}$ has 66 columns and ∼2,000 rows.

ordinary least squares produces the minimum ℓ_2 norm solution (corresponding to a point somewhere in the interior of the feasible set—more on this later when we turn to linear programming), but Theorem 4 indicates that every component will be nonzero, forcing us to turn to the relatively inefficient Steinitz elimination process described in Section 3.3.

An suitable algorithm for solving (31) is the active set method due to Lawson and Hanson [18]. To derive this algorithm, define the Lagrangian of (31):

$$L(\boldsymbol{w}, \boldsymbol{\alpha}) = \frac{1}{2} \left\| \boldsymbol{V}^\top \boldsymbol{w} - \boldsymbol{I} \right\|_2^2 - \boldsymbol{\alpha}^\top \boldsymbol{w}. \tag{32}$$

The Karush-Kuhn-Tucker (KKT) conditions for a constrained local optimum are:

$$L_{\boldsymbol{w}}(\boldsymbol{w}, \boldsymbol{\alpha}) = \boldsymbol{V}\big(\boldsymbol{V}^\top \boldsymbol{w} - \boldsymbol{I}\big) - \boldsymbol{\alpha} = \boldsymbol{0}, \qquad \boldsymbol{\alpha}^\top \boldsymbol{w} = 0, \qquad \boldsymbol{w} \geqslant 0, \qquad \boldsymbol{\alpha} \leqslant 0. \tag{33}$$

In rough outline, an active set method for solving (31) chooses an initial iterate $\boldsymbol{w}_0$ and determines the inactive set $J_0 = \{i : 1 \leqslant i \leqslant n \text{ and } (\boldsymbol{w}_0)_i > 0\}$. After computing $\boldsymbol{\alpha}_0 = \boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{w}_0 - \boldsymbol{I})$, we determine the maximum component of $\boldsymbol{\alpha}_0$, remove it from $\mathcal{I}_0$, and compute $\boldsymbol{w}_1$ by updating the new inactive component of $\boldsymbol{w}_0$. Continuing in this way, $\boldsymbol{w}_k$ and $\boldsymbol{\alpha}_k$ are determined for $k > 1$, until all Lagrange multipliers are nonpositive. Occasionally, it is necessary to backtrack to ensure that the constraints are satisfied exactly for each iteration.

3.5 Accuracy of the Lawson-Hanson algorithm

Lawson and Hanson provide an implementation of an active set method for nonnegative least squares in their classic textbook on solving least squares problems [18]. At a high level, the algorithm follows the approach described in Section 3.4.

For bookkeeping, we use "numpy" notation (i.e., 0-indexed "MATLAB notation"), where, for example, $\boldsymbol{x}_{i:j} = (x_i, x_{i+1}, \ldots, x_j - 1)$; and where expressions like $\boldsymbol{A}_{:,:k}$ work as expected. We also consider expressions with index vector subscripts; e.g., given $I = (i_1, ..., i_n)$, we write $\boldsymbol{x}_I = (x_{i_1}, \ldots, x_{i_n})$. For an index vector $I \subseteq [n]$, we write $I^c \subseteq [n]$ for its complement.

Let:

$$\boldsymbol{R}_k = \left(\boldsymbol{Q}_k^\top \boldsymbol{V}^\top\right)_{:|J_k|,J_k}, \qquad \boldsymbol{b}_k = \left(\boldsymbol{Q}_k^\top \boldsymbol{I}\right)_{:|J_k|} \tag{34}$$

Note that:

$$\left(\boldsymbol{Q}_k^\top \boldsymbol{V}^\top\right)_{|J_k|:,J_k} = \boldsymbol{0}_{(n-|J_k|)\times|J_k|}. \tag{35}$$

At the kth step, we let $J_k \subseteq [n]$ be the set of inactive indices and compute a new tenative value of $\boldsymbol{w}_k$ by solving the least squares problem on the set of inactive indices J_k:

$$\begin{aligned} \boldsymbol{R}_k \left(\boldsymbol{w}_k\right)_{J_k} &= \boldsymbol{b}_k, \\ \left(\boldsymbol{w}_k\right)_{J_k^c} &= \boldsymbol{0}_{|J_k^c|}. \end{aligned} \tag{36}$$

is solved. If any components of $\boldsymbol{w}_k$ are negative, backtracking is used to find the largest step such that $\boldsymbol{w}_k \geqslant 0$. Afterwards, the Lagrange multiplier is recomputed from:

$$\boldsymbol{\alpha}_k = \boldsymbol{V}\left(\boldsymbol{V}^\top \boldsymbol{w}_k - \boldsymbol{I}\right) \tag{37}$$

If $n > m$, the least squares problem (36) will be overdetermined until the last step, when it becomes square.

Lawson and Hanson elect to solve (36) using a QR decomposition. This QR decomposition is updated and downdated on the fly, with columns being added and removed from the orthogonal basis which spans some subset of the columns of $\boldsymbol{V}^\top$. For clarity, note the following:

1) An index becoming *active* (being added to the active set) corresponds to a quadrature node being removed from the rule; equivalently, a column of $\boldsymbol{V}^\top$ being removed from the basis, and an entry of $\boldsymbol{w}$ being reset to zero.
2) An index becoming *inactive* is the same as a new quadrature node being added and an unused column of $\boldsymbol{V}$ being incorporated into the QR decomposition's running orthogonal basis.

Columns are added to the orthogonalized basis using a Householder reflector; they are removed using a Givens rotation. Details of how this is done can be found in their book or elsewhere [**?**]. Both $\boldsymbol{Q}_k^\top \boldsymbol{V}^\top$ and $\boldsymbol{Q}_k^\top \boldsymbol{I}$ are maintained throughout the iteration, where $\boldsymbol{Q}_k \in \mathbb{R}^{m\times m}$ is the product of all Householder reflectors and Givens rotations applied throughout. The matrix $\boldsymbol{Q}_k$ is not stored explicitly. Consequently, the least squares problem (36) reduces to an upper triangular system. No additional pivoting is applied for numerical stability other than what could be regarded as the problem-dependent pivoting induced by the choice of component to add or remove from the active set at each step.

Lawson and Hanson's Fortran implementation is available in the public domain. Until recently, it provided the heavy lifting for `scipy.optimize.nnls`, although it appears to have been replaced by a simpler Python implementation. This is unfortunate, because the original Fortran implementation has an important feature which provides a significant amount of additional stability and accuracy on ill-conditioned problems. In some cases, we found that this minor change allowed the Lawson-Hanson iteration to converge with a residual five orders of magnitude smaller than the naïve implementation. To the best of our knowledge, this detail is not discussed in their book, nor is it explained in a comment in their Fortran implementation. The key detail is this. At each step, the new vector of Lagrange multipliers is computed from:

$$\begin{aligned} \left(\boldsymbol{\alpha}_k\right)_{J_k} &= \boldsymbol{0}_{|J_k|}, \\ \left(\boldsymbol{\alpha}_k\right)_{J_k^c} &= \left[\left(\boldsymbol{Q}_k^\top \boldsymbol{V}^\top\right)^\top\right]_{J_k^c} \boldsymbol{Q}_k^\top \boldsymbol{I}, \end{aligned} \tag{38}$$

The form of (38) allows us to solve for the components of $\boldsymbol{\alpha}_k$ as follows:

1) Set $\left(\boldsymbol{\alpha}_k\right)_J := \boldsymbol{0}_n$.
2) Set $\left(\boldsymbol{\alpha}_k\right)_{J^c} := \left(\boldsymbol{Q}_k^\top \boldsymbol{V}_k^\top\right)^\top_{(|J|+1):} \left(\boldsymbol{Q}_k^\top \boldsymbol{I}\right)_{(|J|+1):}$.

We can see that naïvely computing $\boldsymbol{\alpha}_k$ from (37) requires us to first compute the residual $\boldsymbol{I} - \boldsymbol{V}^\top \boldsymbol{w}_k$. As k increases and the residual reduces in magnitude, this residual computation can sometimes introduce an artificially large "error floor" as we begin "approximating zero" using the differences of moderately numbers. On the other hand, with (38), components of α_k are directly set to zero if they are active, or computed using a triangular solve. In the latter case, the triangular system has been transformed from the original using a sequence of orthogonal transforms, preventing spurious numerical errors from accumulating.

We now justify our claim that Lawson and Hansons's stabilized expression for the Lagrange multipliers given by (38) can be used instead of (37).

Theorem 6. *Expressions* (37) *and* (38) *are equivalent.*

Proof. First, note that since $\boldsymbol{Q}_k$ is a (square) orthogonal matrix, we trivially have:

$$\boldsymbol{\alpha}_k = \boldsymbol{V}\boldsymbol{Q}_k \left(\boldsymbol{Q}_k^\top \boldsymbol{V}^\top \boldsymbol{w}_k - \boldsymbol{Q}_k^\top \boldsymbol{I} \right). \tag{39}$$

We expand the first term in the parenthesis into parts corresponding to J_k and J_k^c and simplify to get:

$$\boldsymbol{Q}_k^\top \boldsymbol{V}^\top \boldsymbol{w}_k = \left(\boldsymbol{Q}_k^\top \boldsymbol{V}^\top \right)_{:,J_k} (\boldsymbol{w}_k)_{J_k} + \left(\boldsymbol{Q}_k^\top \boldsymbol{V}^\top \right)_{:,J_k^c} \underbrace{(\boldsymbol{w}_k)_{J_k^c}}_{=0} = \left(\boldsymbol{Q}_k^\top \boldsymbol{V}^\top \right)_{:,J_k} (\boldsymbol{w}_k)_{J_k}. \tag{40}$$

We can further rewrite the matrix in this last expression as a block matrix to find that:

$$\left(\boldsymbol{Q}_k^\top \boldsymbol{V}^\top \right)_{:,J_k} = \begin{bmatrix} \boldsymbol{R}_k (\boldsymbol{w}_k)_{J_k} \\ \boldsymbol{0}_{(n-|J_k|)\times|J_k|} \end{bmatrix}. \tag{41}$$

Consequently:

$$\boldsymbol{Q}_k^\top \boldsymbol{V}^\top \boldsymbol{w}_k - \boldsymbol{Q}_k^\top \boldsymbol{I} = \begin{bmatrix} \boldsymbol{R}_k (\boldsymbol{w}_k)_{J_k} - \boldsymbol{b}_k \\ -\left(\boldsymbol{Q}_k^\top \boldsymbol{I} \right)_{|J_k|:} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_{|J_k|} \\ -\left(\boldsymbol{Q}_k^\top \boldsymbol{I} \right)_{|J_k|:} \end{bmatrix}. \tag{42}$$

Now, consider $(\boldsymbol{\alpha}_k)_{J_k}$. We have:

$$(\boldsymbol{\alpha}_k)_{J_k} = (\boldsymbol{V}\boldsymbol{Q}_k)_{J_k,:} \begin{bmatrix} \boldsymbol{0}_{|J_k|} \\ -\left(\boldsymbol{Q}_k^\top \boldsymbol{I} \right)_{|J_k|:} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_k^\top & \boldsymbol{0}_{|J_k|\times(n-|J_k|)} \end{bmatrix} \begin{bmatrix} \boldsymbol{0}_{|J_k|} \\ -\left(\boldsymbol{Q}_k^\top \boldsymbol{I} \right)_{|J_k|:} \end{bmatrix} = \boldsymbol{0}_{|J_k|}. \tag{43}$$

For $(\boldsymbol{\alpha}_k)_{J_k^c}$, we can write:

$$(\boldsymbol{\alpha}_k)_{J_k^c} = (\boldsymbol{V}\boldsymbol{Q}_k)_{J_k^c,:} \begin{bmatrix} \boldsymbol{0}_{|J_k|} \\ -\left(\boldsymbol{Q}_k^\top \boldsymbol{I} \right)_{|J_k|:} \end{bmatrix} = -(\boldsymbol{V}\boldsymbol{Q}_k)_{J_k^c,|J_k|:} \left(\boldsymbol{Q}_k^\top \boldsymbol{I} \right)_{|J_k|:}. \tag{44}$$

This proves the claim. □

3.6 Basis pursuit quadrature

A classic model from the compressed sensing literature not too far removed from nonnegative least squares is the *basis pursuit* problem [1]:

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{w}\|_1 \\ \text{subject to} \quad & \boldsymbol{V}^\top \boldsymbol{w} = \boldsymbol{I} \\ & \boldsymbol{w} \geqslant 0 \end{aligned} \tag{45}$$

In the original formulation of basis pursuit, the nonnegativity constraints are optional. We include them in our formulation to present a form of basis pursuit pertinent to the design of positive quadratures. This model was explored by Glaubitz [12].

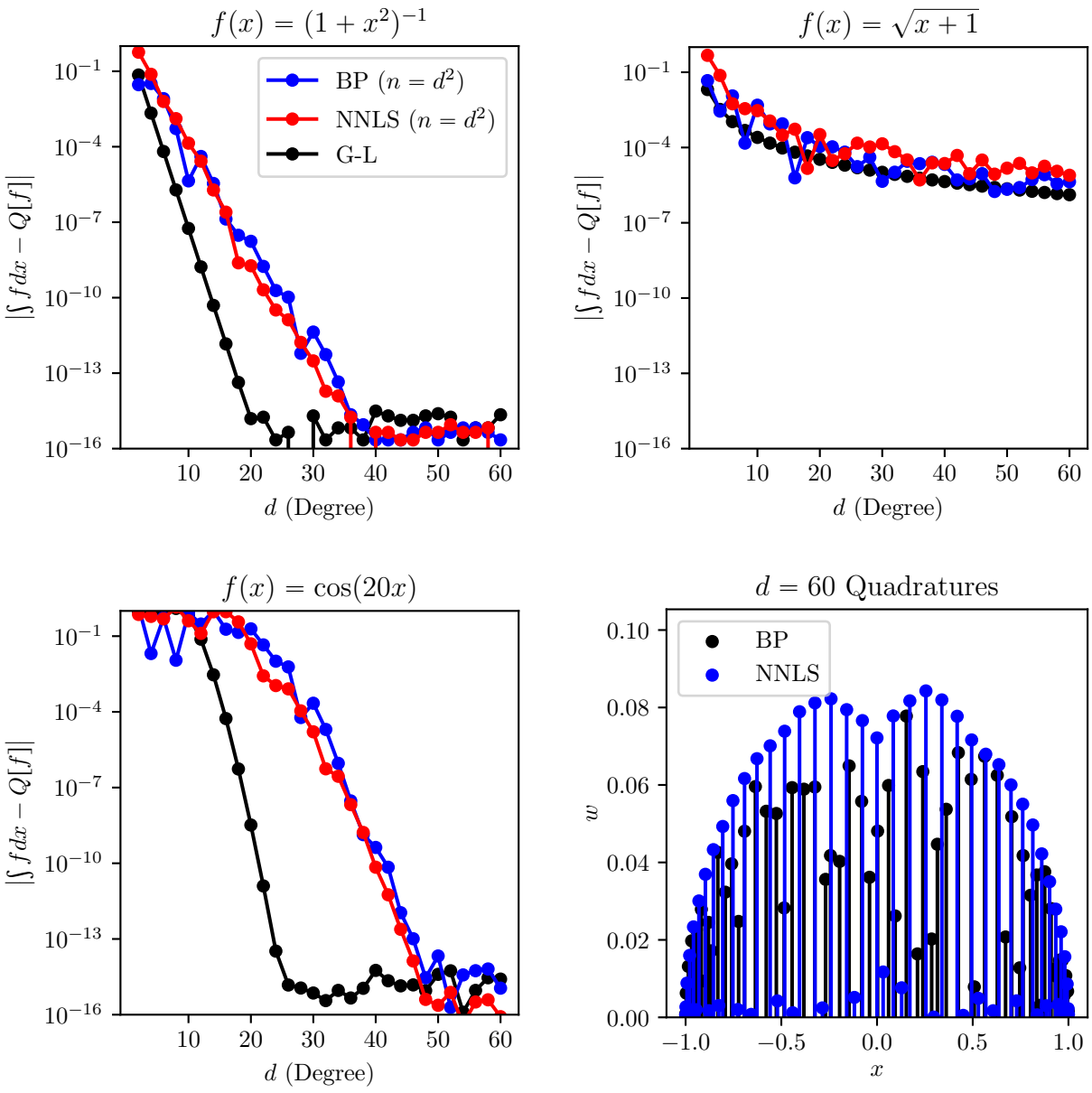

Figure 2: We compare the basis pursuit (BP) quadratures (Section 3.6) and nonnegative least squares (NNLS) quadratures (Section 3.4) with Gauss-Legendre quadratures on the interval for several integrands up to degree 60. To avoid ill-conditioning, we formulate the problem in the Chebyshev basis. We observe that while the NNLS and BP quadratures are comparable in terms of their order of accuracy (being roughly one half of what is achieved by the equivalent Gauss-Legendre rule, as expected), we find that the NNLS rules are more symmetric, with a smooth decay of quadrature weights towards the boundary, as one might expect.

Basis pursuit provides no advantage over nonnegative least squares. The cost function of (45) is piecewise linear. Indeed, since all feasible $\boldsymbol{w}$ are nonnegative, the cost function can be replaced with $\mathbf{1}^\top \boldsymbol{w}$. However, since $\mathbb{P}_N^d$ contains the constant functions, the value of $\mathbf{1}^\top \boldsymbol{w}$ is fixed when the equality constraints $\boldsymbol{V}^\top \boldsymbol{w} - \boldsymbol{I}$ are satisfied since $\mathcal{Q}[1] = \sum_i w_i = \mathbf{1}^\top \boldsymbol{w}$ and since $\mathbf{1}$ can be formed as a linear combination of the columns of $\boldsymbol{V}$. Consequently, we can see that the cost function of (45) can be further simplified to $\mathcal{Q}[1] = \text{vol}(\Omega)$, which is a constant independent of $\boldsymbol{w}$ on the feasible set. Hence, solving (45) with standard linear programming algorithms is equivalent to testing the constraints for feasibility. We can see that choosing a nontrivial cost function then provides an opportunity to incorporate more information into our model (see Section 3.7, where this idea is elaborated upon). In Section 3.6, we demonstrate that the quadratures computed on the interval using basis bursuit and nonnegative least squares are comparable in terms of their accuracy; but we also see that the Lawson-Hanson algorithm produces quadratures which are more symmetric with a smooth roll-off of weights towards the boundary.

A simple example comparing the use of basis pursuit and NNLS for selecting positive quadratures from a uniform grid discretizing $[-1, 1]$ is provided in Section 3.6—we can see that the integration errors are comparable, and the NNLS quadrature even seems to have a more regular node distribution, capturing the symmetry of the domain automatically. Clearly, since basis pursuit reduces to feasibility testing and there is no unique global minimum of the LP, the resulting quadrature is biased by the algorithm used to solve the LP. We do not explore basis pursuit as an algorithm further, other than to note that it is essentially a degenerate form of the LP quadrature proposed by Ryu and Boyd, discussed in Section 3.7.

3.7 Ryu-Boyd quadrature

The only difference between the basis pursuit quadratures (section 3.6) and the "Gauss LP" quadratures due to Ryu and Boyd is the choice of cost function [22]. Let $r(\boldsymbol{x}) : \Omega \to \mathbb{R}$, and let $\boldsymbol{r} = r(\boldsymbol{X})$. Ryu and Boyd refer to this as a *sensitivity function*. The LP for Ryu-Boyd quadrature is:

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{r}^\top \boldsymbol{w}, \\ \text{subject to} \quad & \boldsymbol{V}^\top \boldsymbol{w} = \boldsymbol{I}, \\ & \boldsymbol{w} \geqslant 0. \end{aligned} \tag{46}$$

From this, we can see that basis pursuit is an instance of Ryu-Boyd with $r(\boldsymbol{x}) = 1$. We can conclude that basis pursuit is totally insensitive to the choice of quadrature nodes, since we always consider polynomial spaces that contain constant functions.

What sensitivity function r to use is open-ended—several choices are discussed in the original paper. Clearly, we should choose a function r which doesn't lie in the polynomial space being integrated so that $\boldsymbol{r}$ isn't contained in the column space of $\boldsymbol{V}$ (in which case, every feasible point is globally optimal, as we saw with basis pursuit in the preceding paragraph). Ryu and Boyd prove that if the infinite-dimensional version of (46) is solved in one dimension with $r(x) = x^{2m}$, then its solution recovers the corresponding Gaussian quadrature. They are unable to prove an analogous result in higher dimensions, but they observe experimentally that a choice like $r(x, y) = x^{2m} + y^{2m}$ for the polynomial space $\mathbb{P}_m^2$ leads to good results. They also observe that while choosing $r(x, y) = x^{2m} + y^{2m}$ tends to produce "Gauss" quadratures (i.e., efficient quadratures with strictly interior points), the choice $r(x, y) = -x^{2m} - y^{2m}$ yields "Lobatto" quadratures, where many points from the boundary are selected.

These observations can be contextualized somewhat by considering the convex geometry of the feasible set. Namely, for a fine enough uniform grid sampling Ω, we observe that:

- ordinary least squares produces a fully interior point (Section 3.2),
- nonnegative least squares using the Lawson-Hanson algorithm produces an order m quadrature corresponding to a point on the boundary of the feasible set, lying on a vertex or some facet of the polytope (Section 3.4),
- basis pursuit does likewise, although the difference in algorithmic details biases the result to a different order m quadrature (Section 3.6).

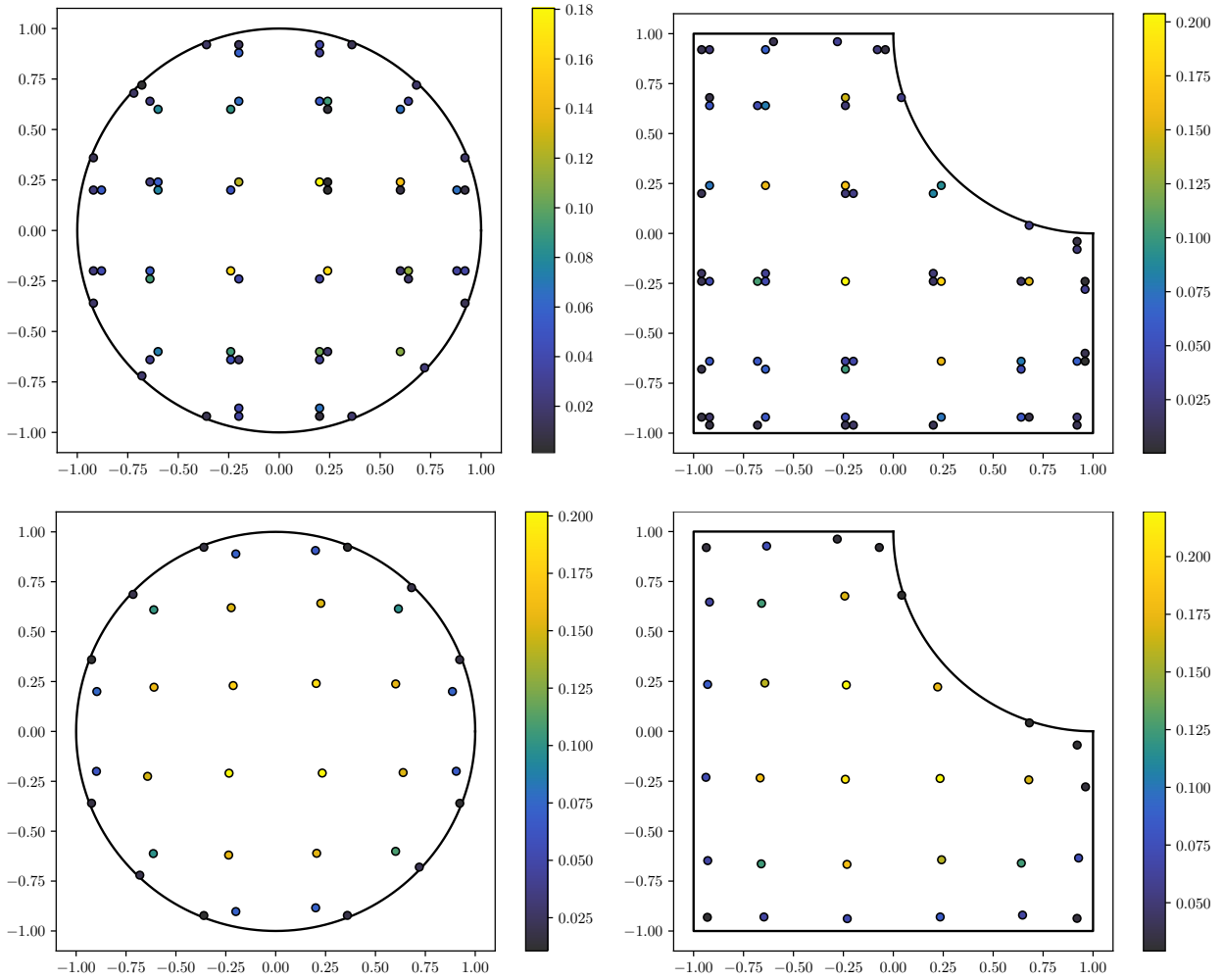

Figure 3: The original Ryu-Boyd algorithm (as described in Section 3.7) applied to the two test problems shown in Figure 1 (the left and right columns in this figure correspond to the left and right columns in Figure 1). The same spatial grid and moment-matching system are used. The residual function $r(x, y) = x^{20} + y^{20}$ is used. *Top*: the LP (46) is solved using the simplex method. The resulting quadrature is shown, exhibiting the effect originally observed by Ryu and Boyd, as small clusters form near ideal node placements. These quadratures integrate $\mathbb{P}_{10}^2$ exactly and have exactly $\dim \mathbb{P}_{10}^2 = 66$ nodes. *Bottom*: The nodes are clustered following Ryu and Boyd's prescription and subsequently optimized using Gauss-Newton. After running Gauss-Newton, all nodes remain in the interior of Ω and all weights remain positive. The left and right column quadratures integrate $\mathbb{P}_{10}^2$ within numerical roundoff and have 36 and 32 nodes, respectively. This is a significant increase in efficiency compared to the NNLS quadratures. Note that the minimum number of nodes possible for these test problems is 22; so, the quadratures have an efficiency of $0.6\overline{1}$ and 0.6875, respectively, while the NNLS quadratures each have an efficiency of 1/3.

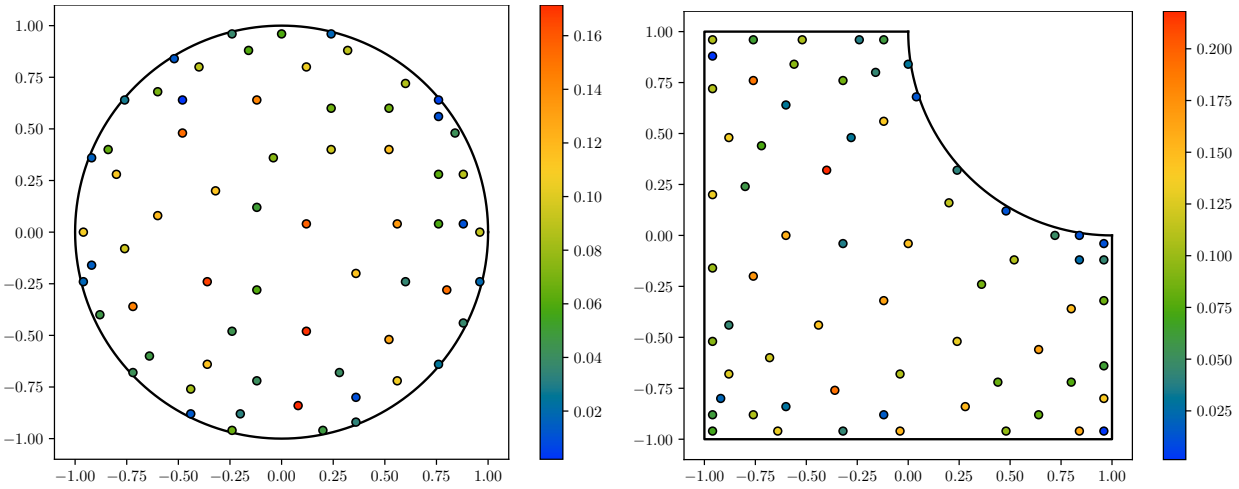

Figure 4: Plots of node significances $w_j\|\boldsymbol{\varphi}(\boldsymbol{x}_j)\|_2$ for the NNLS quadratures computed in Figure 1.

Consequently, it can be seen that introducing the sensitivity function "steers" the optimum of the LP to different quadratures which are at most order m.

A sketch of the Ryu-Boyd algorithm as presented in the original paper follows:

Algorithm 3.3: Sketch of algorithm for computing Ryu-Boyd quadratures

1) Following Section 3.1, let $\boldsymbol{X}$ be a uniform grid sampling Ω (let $h > 0$ be the grid spacing).
2) Solve (46), letting $\boldsymbol{w}$ denote the optimum.
3) Let $\boldsymbol{X}_{\text{LP}} = \{\boldsymbol{x} \in \boldsymbol{X} : \boldsymbol{w}_{\boldsymbol{x}} > 0\}$.
4) Let $\boldsymbol{w}_{\text{LP}} = \{\boldsymbol{w}_{\boldsymbol{x}} : \boldsymbol{x} \in \boldsymbol{X}_{\text{LP}}\}$.
5) Cluster $\boldsymbol{X}_{\text{LP}}$ to obtain $\boldsymbol{X}_{\text{clust}}$ and $\boldsymbol{w}_{\text{clust}}$. This is heuristic. As an example, we could call two points $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}$ "neighbors" if $\|\boldsymbol{x} - \boldsymbol{y}\|_\infty \leqslant h$ (giving $3^d - 1$ point neighborhoods). Finding all connected components in the resulting graph gives a connected component for each cluster: to find $\boldsymbol{x} \in \boldsymbol{X}_{\text{clust}}$, compute their mean. To obtain $\boldsymbol{w}_{\text{clust}}$, sum together the weights of each node in the connected component (this maintains $\sum_{\boldsymbol{x} \in \boldsymbol{X}_{\text{clust}}} \boldsymbol{w}_{\boldsymbol{x}} = \text{vol}(\Omega)$).
6) Use Gauss-Newton to optimize the moment-matching equations with $(\boldsymbol{w}_{\text{clust}}, \boldsymbol{X}_{\text{clust}})$ as a warm start—denote the result by $(\boldsymbol{w}_{\text{opt}}, \boldsymbol{X}_{\text{opt}})$.
7) The quadrature determined by $(\boldsymbol{w}_{\text{opt}}, \boldsymbol{X}_{\text{opt}})$ is the result.

If h is too large, this algorithm can fail in two ways. First, clusters might begin to overlap or connect, preventing us from determining the location of all quadrature nodes. Second, the Gauss-Newton iteration may diverge if $(\boldsymbol{w}_{\text{clust}}, \boldsymbol{X}_{\text{clust}})$ is an insufficiently good warm start. Both of these problems are mitigated by letting $h \to 0$, giving us some confidence that this algorithm is consistent.

3.8 Xiao-Gimbutas elimination

Steinitz elimination (Section 3.3) allows us to reduce an order $n > m$ positive quadrature to an order $n = m$ positive quadrature. If $\mathcal{Q}$ is the Steinitz-eliminated quadrature, then $\text{eff}(\mathcal{Q}) = \frac{1}{d+1}$. Since examples abound of positive quadratures in d dimensions with $\text{eff} > \frac{1}{d+1}$, it is natural to ask whether we can further reduce the quadrature by eliminating nodes in some other way. Sadly, little is known in general about optimal multivariate quadratures, so it is unclear what our notion of optimality should be, other than driving $\text{eff}(\mathcal{Q}) \leqslant 1$ as high as possible. Nevertheless, Xiao and Gimbutas provide a method of eliminating nodes using a nonlinear procedure [27]. We summarize their approach to node elimination here.

Unit disk							Square with circular bite						
d	$\varphi_{\max}$	σ_j	$\lvert\mathbb{P}_d^2\rvert$	$\lvert\mathcal{X}\rvert$	Eff	G-N	d	$\varphi_{\max}$	σ_j	$\lvert\mathbb{P}_d^2\rvert$	$\lvert\mathcal{X}\rvert$	Eff	G-N
8	-0.003	Res	45	37	*40.5%*	5	8	-0.003	Res	45	43	34.9%	5
8	-0.003	XG1	45	44	34.1%	8	5	-0.003	XG1	45	42	**35.7%**	7
8	-0.003	XG2	45	37	*40.5%*	5	8	-0.003	XG2	45	43	34.9%	5
8	-0.01	Res	45	35	42.9%	5	8	-0.01	Res	45	35	42.9%	6
8	-0.01	XG1	45	44	34.1%	5	8	-0.01	XG1	45	42	35.7%	7
8	-0.01	XG2	45	34	**44.1%**	5	8	-0.01	XG2	45	34	**44.1%**	6
8	-0.03	Res	45	32	46.9%	6	8	-0.03	Res	45	29	*51.7%*	10
8	-0.03	XG1	45	44	34.1%	5	8	-0.03	XG1	45	42	35.7%	7
8	-0.03	XG2	45	30	**50.0%**	5	8	-0.03	XG2	45	29	*51.7%*	10
10	-0.003	Res	66	56	*39.3%*	5	10	-0.003	Res	66	58	**37.9%**	5
10	-0.003	XG1	66	63	34.9%	8	10	-0.003	XG1	66	61	36.1%	8
10	-0.003	XG2	66	56	*39.3%*	5	10	-0.003	XG2	66	60	36.7%	5
10	-0.01	Res	66	54	40.7%	5	10	-0.01	Res	66	56	39.3%	6
10	-0.01	XG1	66	63	34.9%	7	10	-0.01	XG1	66	64	34.4%	6
10	-0.01	XG2	66	52	**42.3%**	5	10	-0.01	XG2	66	54	**40.7%**	6
10	-0.03	Res	66	48	*45.8%*	7	10	-0.03	Res	66	47	46.8%	8
10	-0.03	XG1	66	63	34.9%	6	10	-0.03	XG1	66	62	35.5%	8
10	-0.03	XG2	66	48	*45.8%*	7	10	-0.03	XG2	66	45	**48.9%**	9
12	-0.003	Res	91	85	36.5%	4	12	-0.003	Res	91	79	**39.2%**	7
12	-0.003	XG1	91	90	34.4%	5	12	-0.003	XG1	91	90	34.4%	6
12	-0.003	XG2	91	78	39.7%	6	12	-0.003	XG2	91	84	36.9%	6
12	-0.01	Res	91	80	38.8%	7	12	-0.01	Res	91	76	*40.8%*	7
12	-0.01	XG1	91	90	34.4%	5	12	-0.01	XG1	91	88	35.2%	7
12	-0.01	XG2	91	78	**39.7%**	5	12	-0.01	XG2	91	76	*40.8%*	5
12	-0.03	Res	91	63	49.2%	7	12	-0.03	Res	91	65	47.7%	11
12	-0.03	XG1	91	89	34.8%	6	12	-0.03	XG1	91	88	35.2%	9
12	-0.03	XG2	91	62	**50.0%**	7	12	-0.03	XG2	91	59	**52.5%**	13

Table 1: For our two test domains, and for a variety of degrees, we run Xiao-Gimbutas elimination until failure for each node significance. We find that the ℓ_2-based significance and XG2 perform the best and are routinely capable of boosting the efficiency of each quadrature by 10-15%.

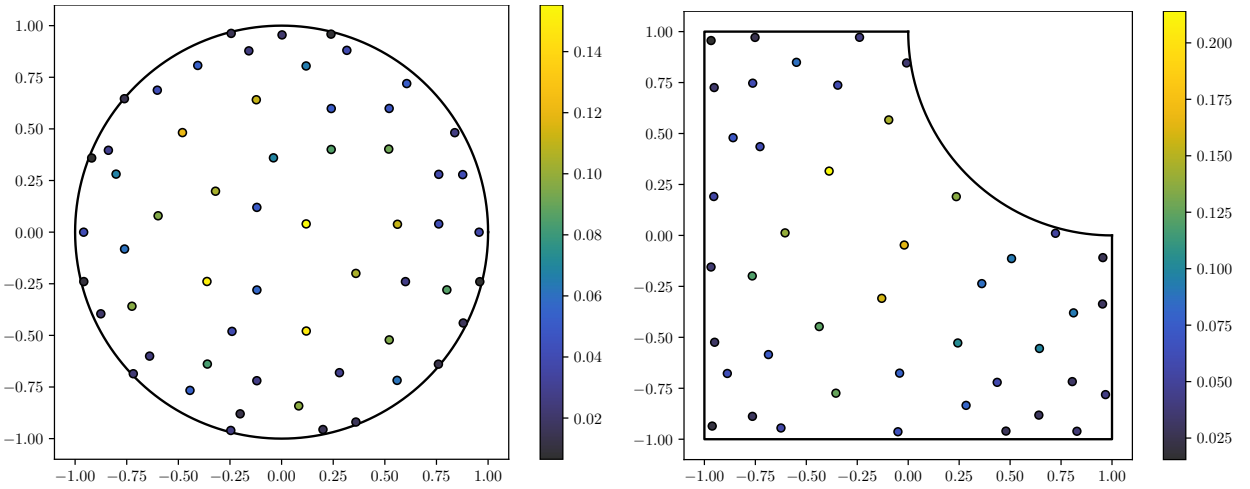

Figure 5: NNLS quadratures from Figure 1 after using Xiao-Gimbutas elimination to remove one node at a time until Gauss-Newton fails to converge. In each case, we can see that this process is able to remove a significant number of nodes.

Xiao and Gimbutas define the *significance* of a quadrature node to be either:

$$\sigma_j^{\mathrm{XG1}} = \sum_{i=1}^{m} \varphi_i(\boldsymbol{x}_j)^2 \qquad \text{or} \qquad \sigma_j^{\mathrm{XG2}} = w_j \sum_{i=1}^{m} \varphi_i(\boldsymbol{x}_j)^2. \tag{47}$$

Intuitively, we can see that σ_j will be large if it contributes a significant amount to the moments $I_1, \ldots, I_m$, and small otherwise. Xiao and Gimbutas eliminate quadrature nodes one at a time by first constructing a new quadrature by leaving out the *least significant* quadrature node (the node with the smallest value of σ) and then re-optimizing using Gauss-Newton. Their goal is to design quadratures for canonical domains which are as close to optimal ($\mathrm{eff}(\mathcal{Q}) = 1$) as possible, and they do achieve this end. For such domains, their quadratures are likely the best that exist in many cases. Crucially, their approach involves recursively backtracking over possible node elimination sequences, keeping the most efficient quadrature found. Their recursion terminates if Gauss-Newton diverges or if it converges to a quadrature with negative weights or nodes outside Ω. (Alternatively, one might consider using an algorithm for inequality-constrained nonlinear optimization (e.g. sequential quadratic programming [3, 21]); likely, such an implementation would be more efficient (possibly obviating the need for recursive backtracking), but would come at the cost of significant implementation complexity.)

To motivate Xiao and Gimbutas' notion of significance, let $\mathcal{Q}$ be a quadrature with weights $w_1, \ldots, w_n$ and nodes $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ which integrates $\mathcal{P} = \mathrm{span}\{\varphi_1, \ldots, \varphi_m\}$, and let $\tilde{\mathcal{Q}}$ denote the quadrature obtained by removing the jth node from $\mathcal{Q}$. Let $\boldsymbol{V}$ and $\boldsymbol{w}$ be the Vandermonde matrix and quadrature weight vector associated with $\mathcal{Q}$; likewise, define $\tilde{\boldsymbol{V}}$ and $\tilde{\boldsymbol{w}}$ to go along with $\tilde{\mathcal{Q}}$. As usual, let $\boldsymbol{I}$ be the moment vector. Since $\mathcal{Q}$ integrates $\mathcal{P}$, the moment-matching equations are satisfied: $\boldsymbol{V}^\top \boldsymbol{w} = \boldsymbol{I}$. Define $\boldsymbol{\varphi}(\boldsymbol{x}) = (\varphi_1(\boldsymbol{x}), \ldots, \varphi_m(\boldsymbol{x}))$. Note that with $\boldsymbol{\varphi}(\boldsymbol{x})$ so defined, the Xiao-Gimbutas significances become $\sigma_j^{\mathrm{XG1}} = \|\boldsymbol{\varphi}(\boldsymbol{x}_j)\|_2^2$ and $\sigma_j^{\mathrm{XG2}} = w_j \|\boldsymbol{\varphi}(\boldsymbol{x}_j)\|_2^2$, respectively. From

$$\boldsymbol{I} = \boldsymbol{V}^\top \boldsymbol{w} = \sum_{j=1}^{n} w_j \boldsymbol{\varphi}(\boldsymbol{x}_j) = w_j \boldsymbol{\varphi}(\boldsymbol{x}_j) + \sum_{\substack{k=1 \\ k \neq j}}^{n} w_k \boldsymbol{\varphi}(\boldsymbol{x}_k) = w_j \boldsymbol{\varphi}(\boldsymbol{x}_j) + \tilde{\boldsymbol{V}}^\top \tilde{\boldsymbol{w}}, \tag{48}$$

we have $\boldsymbol{I} - \tilde{\boldsymbol{V}}^\top \tilde{\boldsymbol{w}} = w_j \boldsymbol{\varphi}(\boldsymbol{x}_j)$. That is, we can see that $w_j \boldsymbol{\varphi}(\boldsymbol{x}_j)$ is exactly the residual of the moment-matching equations for $\tilde{\mathcal{Q}}$. Clearly, the closer either of the two Xiao-Gimbutas significances are to zero, the closer the moment-matching equations for $\tilde{\mathcal{Q}}$ are to being satisfied.

The full Xiao-Gimbutas algorithm (including recursive backtracking) is too costly to run on the fly. However, to take advantage of their method of eliminating nodes in the context of quickly generating quadrature rules for many different unstructured cut cells, we can try speculatively eliminating nodes.

Algorithm 3.4: Simple greedy algorithm for node elimination

1) Compute $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)$ for some definition of σ_j.
2) Remove the jth node, where $j = \arg\min_j \sigma_j$.
3) Use Gauss-Newton to reoptimize the quadrature by solving the moment-matching equations.
4) Repeat this process while all weights are positive and all nodes lie within Ω.

Clearly, the sequence of eliminated nodes depends on the definition of σ_j.

We collect some numerical experiments showing the result of pruning different NNLS quadratures in Table 1. We note that the Xiao-Gimbutas significances are not *exactly* the desired residual. To this end, we include another node significance in our experiment, based more directly on the "punctured" moment-matching residual:

$$\sigma_j^{\text{Res}} = w_k \|\boldsymbol{\varphi}(\boldsymbol{x}_j)\|_2. \tag{49}$$

We found that σ_j^{Res} performs similarly to XG2. However, In practice, we found that σ_j^{XG1} fares quite poorly, typically failing to eliminate very many nodes at all.

Reoptimizing the quadrature rule using Gauss-Newton after eliminating a node is quite cheap compared to solving the original NNLS problem. The underdetermined nonlinear least squares problem that needs to be solved has $m = |\mathbb{P}_n^2|$ equations and $(d+1)|\mathcal{X}| = O(m)$ unknowns. At the beginning of running the greedy algorithm, the eliminated quadratures will tend to be very good warm starts for the Gauss-Newton iteration. Quadratic convergence is typically attained, with Gauss-Newton converging to numerical roundoff in 5 or 6 iterations (for reference, the maximum number of Gauss-Newton iterations are reported in Table 1). The initial NNLS quadrature will have m nodes, and the minimum possible is $\lceil m/(d+1) \rceil$. Conservatively, if we assume the cost of each elimination is $O(m^3)$, then the overall cost of running the simple greedy algorithm for elimination is $O(m^4)$.

Adapting the NNLS discretization for Xiao-Gimbutas elimination. If the bounding box of Ω is tight and we have very many grid points on Ω's boundary, the Lawson-Hanson minimizer will tend to include points along the boundary (the Lawson-Hanson quadratures tend to be Lobatto quadratures). Deleting a node from one of these Lobatto quadratures and reoptimizing will spread the remaining points out slightly, pushing some boundary points outside of Ω, thereby violating one of our desired constraints. To avoid this problem, when sampling our grid, we can choose a value $\varphi_{\max}$ slightly less than zero and only keep grid points $\boldsymbol{x}$ which satisfy $\varphi(\boldsymbol{x}) \leqslant \varphi_{\max}$ to avoid an initial grid with boundary points. With this modification, as our results in Table 1 show, Xiao-Gimbutas can often be run for many steps before any nodes exit the domain, allowing us to significantly improve the efficiency of the Lawson-Hanson quadratures.

4 Fast Ryu-Boyd quadrature

We have not yet addressed how to approach solving the LP eq. (46) involved in computing a Ryu-Boyd quadrature rule. Analogous to the Lawson-Hanson algorithm used to solve the nonnegative least squares problem (31), we can use any off-the-shelf implementation of the simplex method. Alternatively, we could use an interior point method. While the simplex method reveals information about individual quadrature points as it proceeds, interior point methods compute iterates along a central path in the interior of the feasible set which converge to the optimal solution (which is unique if the residual function is chosen appropriately—see Section 3.6). Consequently, we might anticipate gaining information about all quadrature points simultaneously as we run an interior point method; and, indeed, this is the case. See, e.g., Figure 7. It is clear that iterating an interior point LP solver has the effect of concentrating the initial iterate's initially spread out measure closer and closer around the final quadrature points until convergence is reached. In any case,

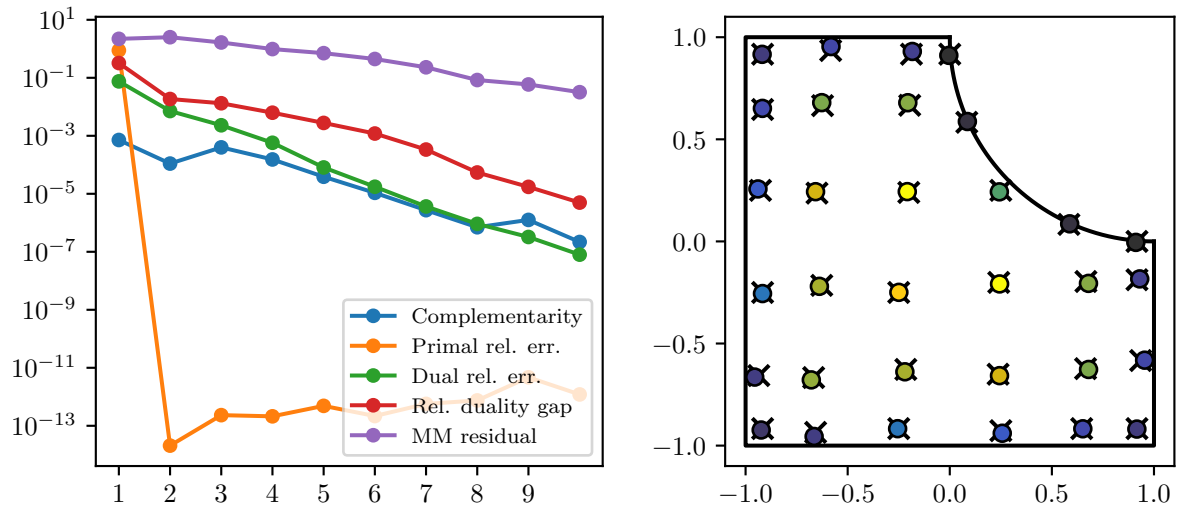

Figure 6: *Left*: TODO. *Right*: the "clustered" quadrature (×'s), and the optimized quadrature (○'s). All nodes in both the clustered quadrature and subsequent optimized quadrature are strictly interior to Ω, and the weights are positive (lying in a range of about 0.014 to 0.239, the exact values being unimportant).

the clustering of several nonzero weights on the uniform grid surrounding the final quadrature points (as shown in Figure 3) is observed by either algorithm since they converge to the same (generally unique) global minimum of (46).

In this section we study one possible realization of an interior point method that is quite efficient. We start from a standard primal-dual interior point method ("Algorithm MPC") presented by Wright [26], which is a particular instance of Mehrotra's predictor-corrector method [20]. Some details are omitted, which can be found in the original reference:

Algorithm 4.1: Using Algorithm MPC to solve (46)

1) Compute the LU decomposition of $\boldsymbol{V}^\top \boldsymbol{V}$.
2) Set $\boldsymbol{w}_0 = \boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{V})^{-1} \boldsymbol{I}$.
3) Set $\boldsymbol{\lambda}_0 = (\boldsymbol{V}^\top \boldsymbol{V})^{-1} \boldsymbol{V}^\top \boldsymbol{r}$.
4) Set $\boldsymbol{s}_0 = \boldsymbol{r} - \boldsymbol{V}\boldsymbol{\lambda}_0$.
5) Shift $\boldsymbol{w}_0$ and $\boldsymbol{s}_0$ by a constant vector so that $\boldsymbol{w}_0 > 0$ and $\boldsymbol{s}_0 > 0$.
6) For $k = 0, 1, \ldots$ (until convergence):
   a) Solve:

$$\begin{bmatrix} & \boldsymbol{V} & \mathsf{Id} \\ \boldsymbol{V}^\top & & \\ \operatorname{diag}(\boldsymbol{s}_k) & & \operatorname{diag}(\boldsymbol{w}_k) \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{w}_k^{\text{aff}} \\ \Delta \boldsymbol{\lambda}_k^{\text{aff}} \\ \Delta \boldsymbol{s}_k^{\text{aff}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r} - \boldsymbol{s}_k - \boldsymbol{V}\boldsymbol{\lambda}_k \\ \boldsymbol{I} - \boldsymbol{V}^\top \boldsymbol{w}_k \\ \operatorname{diag}(\boldsymbol{s}_k) \operatorname{diag}(\boldsymbol{w}_k) \mathbb{1} \end{bmatrix} \tag{50}$$

   b) Set $\alpha_{\text{aff}}^{\text{primal}} = \max(\alpha : 0 \leqslant \alpha \leqslant 1, \boldsymbol{w}_k + \alpha \Delta \boldsymbol{w}_k^{\text{aff}} \geqslant 0)$.
   c) Set $\alpha_{\text{aff}}^{\text{dual}} = \max(\alpha : 0 \leqslant \alpha \leqslant 1, \boldsymbol{s}_k + \alpha \Delta \boldsymbol{s}_k^{\text{aff}} \geqslant 0)$.
   d) Set $\mu_{\text{aff}} = (\boldsymbol{w}_k + \alpha_{\text{aff}}^{\text{primal}} \Delta \boldsymbol{w}_k^{\text{aff}})^\top (\boldsymbol{s}_k + \alpha_{\text{aff}}^{\text{primal}} \Delta \boldsymbol{s}_k^{\text{aff}})/n$.
   e) Set $\mu_k = \boldsymbol{w}_k^\top \boldsymbol{s}_k / n$.
   f) Set $\sigma_k = (\mu_{\text{aff}}/\mu_k)^3$.

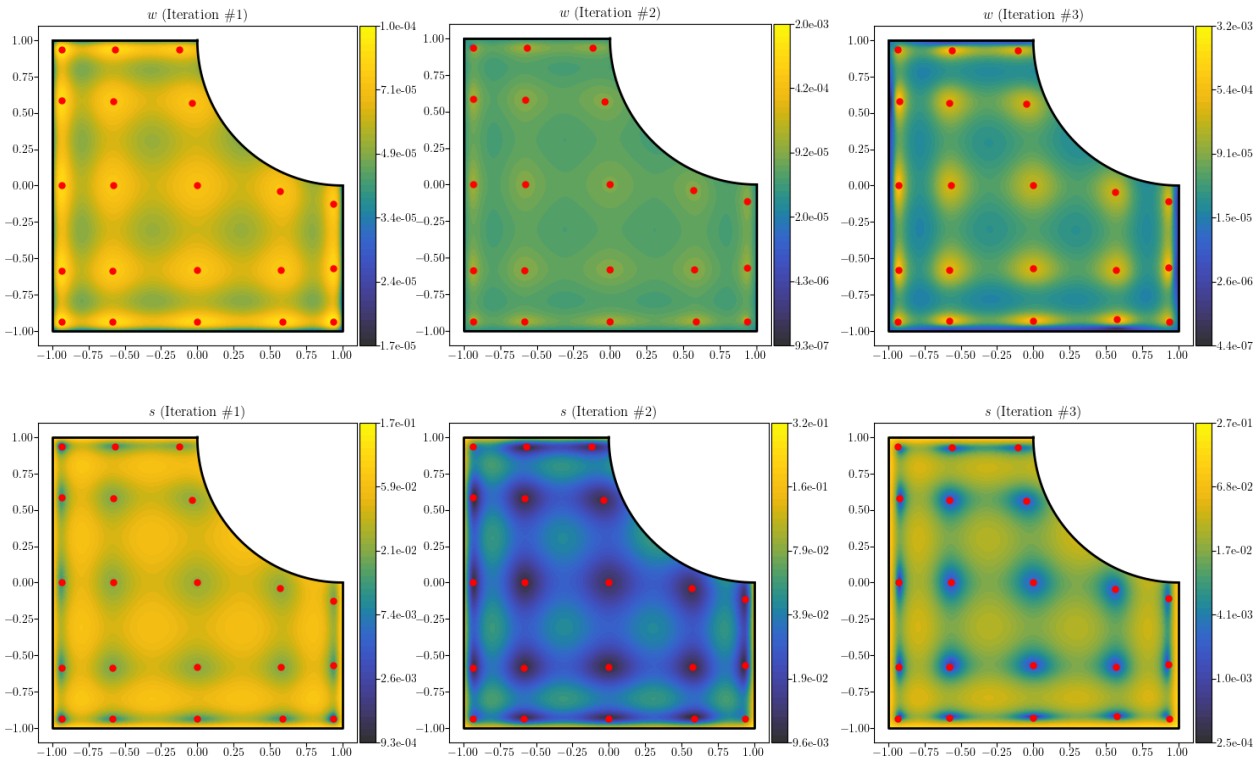

Figure 7: For the same test problem as in the left column of Figure 1 and Figure 3, we plot the first three iterations' primal and dual variables for the predictor-corrector interior point method applied to the Ryu-Boyd quadrature LP. *Left to right*: the iterations of the interior point method. *Top*: the primal variables. *Bottom*: the dual variables. For each iteration, the local maximum clustering is applied to produce a warm start for Gauss-Newton used to solve the moment-matching equations (all Gauss-Newton iterations immediately achieve quadratic converge and produce quadratures with a residual on the order of machine precision). The red dots denote locations of local maxima.

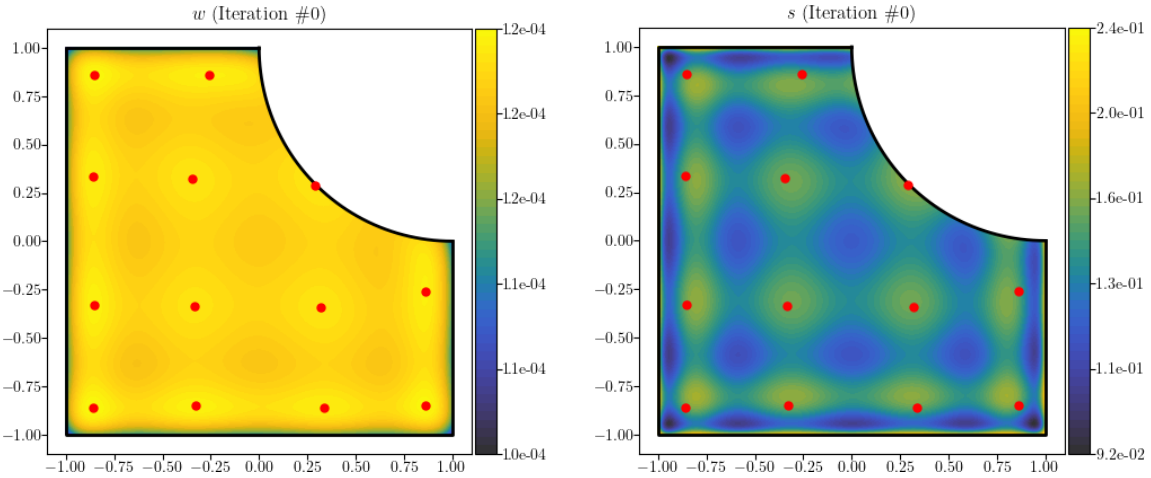

Figure 8: The leftmost column ("Iteration #1") is the warm start, which is essentially a least squares quadrature (see **??**). The quadrature computed this way is clearly incorrect for Iteration #0. However, it is interesting to observe that the interior local minima of the dual variable at Iteration #0 correspond to the correct quadrature.

g) Solve:

$$\begin{bmatrix} & \boldsymbol{V} & \mathsf{Id} \\ \boldsymbol{V}^\top & & \\ \operatorname{diag}(\boldsymbol{s}_k) & & \operatorname{diag}(\boldsymbol{w}_k) \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{w}_k^{(\mathrm{cc})} \\ \Delta \boldsymbol{\lambda}_k^{(\mathrm{cc})} \\ \Delta \boldsymbol{s}_k^{(\mathrm{cc})} \end{bmatrix} = \begin{bmatrix} \\ \sigma_k \mu_k \mathbb{1} - \operatorname{diag}(\Delta \boldsymbol{w}_k^{\mathrm{aff}}) \operatorname{diag}(\boldsymbol{s}_k^{\mathrm{aff}}) \mathbb{1} \end{bmatrix} \tag{51}$$

h) Set $\Delta \boldsymbol{w}_k = \Delta \boldsymbol{w}_k^{\mathrm{aff}} + \Delta \boldsymbol{w}_k^{(\mathrm{cc})}$.
i) Set $\Delta \boldsymbol{\lambda}_k = \Delta \boldsymbol{\lambda}_k^{\mathrm{aff}} + \Delta \boldsymbol{\lambda}_k^{(\mathrm{cc})}$.
j) Set $\Delta \boldsymbol{s}_k = \Delta \boldsymbol{s}_k^{\mathrm{aff}} + \Delta \boldsymbol{s}_k^{(\mathrm{cc})}$.
k) Set $\alpha_{\max}^{\mathrm{primal}} = \max(\alpha : \alpha \geqslant 0, \boldsymbol{w}_k + \alpha \Delta \boldsymbol{w}_k \geqslant 0)$.
l) Set $\alpha_{\max}^{\mathrm{dual}} = \max(\alpha : \alpha \geqslant 0, \boldsymbol{s}_k + \alpha \Delta \boldsymbol{s}_k \geqslant 0)$.
m) Set $\alpha_k^{\mathrm{primal}} = \min(0.99 \cdot \alpha_{\max}^{\mathrm{primal}}, 1)$.
n) Set $\alpha_k^{\mathrm{dual}} = \min(0.99 \cdot \alpha_{\max}^{\mathrm{dual}}, 1)$.
o) Set $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k^{\mathrm{primal}} \Delta \boldsymbol{w}_k$.
p) Set $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha_k^{\mathrm{dual}} \Delta \boldsymbol{\lambda}_k$.
q) Set $\boldsymbol{s}_{k+1} = \boldsymbol{s}_k + \alpha_k^{\mathrm{dual}} \Delta \boldsymbol{s}_k$.

The algorithm is complicated, but the rough idea is to take steps in the Newton direction for the LP's KKT system, while scaling and backtracking to ensure that the trajectory follows the central path. We can also immediately observe that the initialization phase sets $\boldsymbol{w}_0$ to the least squares quadrature.

We plot several iterates of Algorithm 4.1 in Figure 7. From these plots, we can see straightaway that that the clusters found in Figure 3 coincide with the local extrema of the Lagrange multiplier fields. We can run Algorithm 4.1 until it converges, but this can easily take 10+ steps. As an alternative, we can try finding the local extrema of one of the Lagrange multiplier fields and using those nodes as a warm start for Gauss-Newton optimization. For example, an algorithm based on finding local maxima of some iterate $\boldsymbol{w}_k$ might look like:

Algorithm 4.2: Heuristic primal solver for computing Ryu-Boyd quadratures

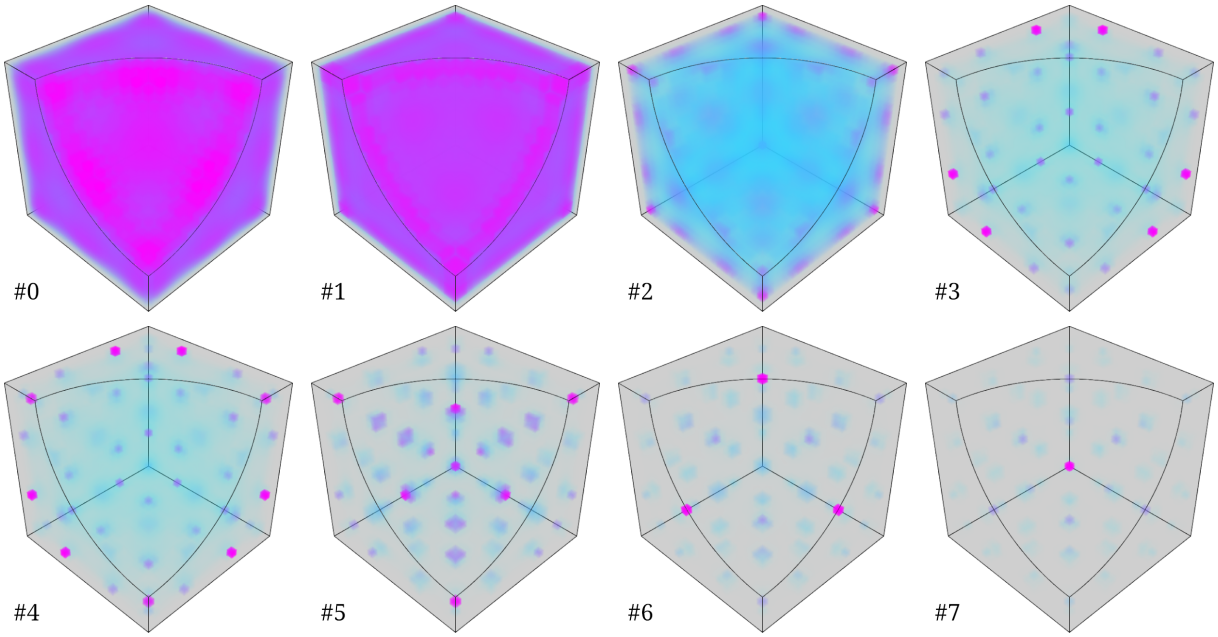

Figure 9: *From left to right, top to bottom*: the warm start primal variables for the interior point method (the least squares quadrature for the grid), followed by the first seven interior point primal iterations. The domain is $\Omega = [-1,1]^3 - B_{3/2}((1,1,1))$. Despite appearances, after the first iteration, the local maxima of the primal variable give an initial iterate for Gauss-Newton which immediately leads to quadratic convergence when solving the moment matching equations. We neglect to plot the scale of the primal variables here (the color is scaled to the range $[0, \max(w_k)]$ in each subplot, where w_k is the primal variable field at the kth iteration, with $0 \leqslant k < 8$). The purpose of the plot is to drive home the point that each successive interior point iteration serves to concentrate the measure nearer and nearer to the true quadrature points (which are almost surely not grid points). The key observation is that we can already deduce the location of each quadrature point from early iterations by looking at local maxima of the primal variable field.

1) Run Algorithm 4.1 to compute $\boldsymbol{w}_k$ for $k > 0$ small; say, $k = 1$ or 2.
2) Find all strict local maxima of $\boldsymbol{w}_k$.
3) For the grid nodes corresponding to the local maxima, run Algorithm 3.2 to compute quadrature weights (they will generally be positive).
4) Run Gauss-Newton to optimize the result.

Algorithm 3.2 is used to compute $\boldsymbol{w}_0$ within Algorithm 4.1 to initialize it. From Figure 8, it is clear that the local maxima of $\boldsymbol{w}_0$ do not coincide with the locations of the final quadrature nodes. So, it is necessary to run Algorithm 4.1 for at least one step. However, it is interesting to observe that the local minima of $\boldsymbol{s}_0$ already correspond quite well to the final quadrature nodes. Even more interesting is that the manner in that $\boldsymbol{s}_0$ has a straightforward interpretation.

Let's see how to interpret each of the dual variables in Algorithm 4.1. The dual variables $\boldsymbol{\lambda}_k$ can be regarded as coefficients of a polynomial in $\mathbb{P}$, and multiplication by $\boldsymbol{V}$ evaluates a polynomial with a given coefficient vector at each grid point. We know already that $\boldsymbol{r}$ consists of the samples of a function on our uniform grid on Ω. Since the columns of $\boldsymbol{V}$ are the basis functions sampled at the grid points, the matrix $\boldsymbol{V}^\top \boldsymbol{V}$ approximates the L^2 Gram matrix, suitably normalized:

$$h^d \left(\boldsymbol{V}^\top \boldsymbol{V} \right)_{ij} = \sum_k h^d \varphi_i(\boldsymbol{x}_k) \varphi_j(\boldsymbol{x}_k) \to (\varphi_i, \varphi_j)_{L^2(\Omega)} \text{ as } h \to 0. \tag{52}$$

Iter.	$\kappa(\mathsf{D}F)$	$\|R_0\|_\infty$	$\|R_{\text{final}}\|_\infty$	$\|X\|$	Eff.	#GN
1	9.4e+02	5.3e-02	1.3e-15	52	40.4%	5
2	9.0e+02	5.9e-02	8.9e-16	50	42.0%	5
3	9.2e+02	5.8e-02	2.7e-15	47	44.7%	5
4	8.7e+02	3.1e-02	4.4e-16	46	45.7%	5
5	9.4e+02	2.8e-02	1.1e-15	44	**47.7%**	5
6	8.3e+02	2.0e-02	8.9e-16	56	37.5%	5
7	8.8e+02	1.8e-02	3.6e-15	55	38.2%	5
8	9.0e+02	1.8e-02	8.9e-16	55	38.2%	5

Table 2: For the same setup as Figure 9, some statistics for the primal-dual iterations. These statistics particularly concern the selected warm start (the "clustered" quadrature) and the subsequent Gauss-Newton iteration for solving the moment-matching equations. The statistics show the "quality" of the warm start is only marginally improved with additional primal-dual iterations. The efficiency of the quadrature improves somewhat after several iterations (see **bold**), but begins to degrade beyond that point. *From left to right*: the primal-dual iteration ("Iter."), the condition number of the Jacobian of moment-matching residuals evaluated at the warm start ("$\kappa(\mathsf{D}F)$"), the max norm of the moment-matching residual for the warm start ("$\|R_0\|_\infty$"), the max norm of the final moment-matching residual ("$\|R_{\text{final}}\|_\infty$"), the number of quadrature points ("$|X|$"), the efficiency of the quadrature ("Eff."), and the number of Gauss-Newton iterations needed for convergence ("#GN").

, and multiplying a vector by $h^d \boldsymbol{V}^\top$ computes an approximation of the dot products with each of the basis functions, e.g. $h^d(\boldsymbol{V}^\top \boldsymbol{r})_i \to (\varphi_i, r)_{L^2(\Omega)}$ as $h \to 0$. Hence, setting $\boldsymbol{\lambda}_0 = (\boldsymbol{V}^\top \boldsymbol{V})^{-1} \boldsymbol{V}^\top \boldsymbol{r}$ is an orthogonal projection of the function r onto the space $\mathbb{P}_N^d$ with respect to the uniform discrete measure supported on our grid sampling Ω. Then, $\boldsymbol{s}_0 = \boldsymbol{r} - \boldsymbol{V}\boldsymbol{\lambda}_0$ consists of the samples of vector rejection of r from $\mathbb{P}_N^d$. Since this uniform discrete measure approximates the (continuous) uniform measure on Ω as $h \to 0$, this is a consistent approximation of the L^2 projection.

Algorithm 4.3: Heuristic dual solver for computing Ryu-Boyd quadratures

1) Compute $\boldsymbol{s} = \boldsymbol{r} - \boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{V})^{-1} \boldsymbol{V}^\top \boldsymbol{r}$.
2) Find all strict local minima of $\boldsymbol{s}$.
3) Optimize the result using Gauss-Newton, as in Algorithm 4.2.

In the continuous setting, if $\{\varphi_1, \ldots, \varphi_m\}$ is a basis for polynomial space $\mathbb{P}$ and $r \in L^2(\Omega)$ is the underlying function from which $\boldsymbol{r}$ is sampled, then we can compute s in much the same way as in the first step of Algorithm 4.3 as follows. First compute $\boldsymbol{b}_i = (\varphi_i, r)_{L^2(\Omega)}$ for each i. Then, form the Gram matrix $\boldsymbol{G}_{ij} = (\varphi_i, \varphi_j)_{L^2(\Omega)}$ and set $\boldsymbol{c} = \boldsymbol{G}^{-1}\boldsymbol{b}$. Finally, set $s = r - \sum_i \boldsymbol{c}_i \varphi_i$ (as $h \to 0$, $\boldsymbol{s}$ computed in Algorithm 4.3 converges to s). Using the continuous formulation, we can gather some evidence that Algorithm 4.3 is a credible algorithm.

Example 4.1. Let $\Omega = [-1, 1]$ and let $r(x) = x^{2n}$. Let $\varphi_i(x) = x^i$ for i such that $0 \leqslant i < 2n$. We find that $s(x) = P_{2n}(x)$, the $2n$th Legendre polynomial. To find the local minima of $P_{2n}(x)$, we consider every other zero of P'_{2n}. Call these zeros x_j $(j = 1, \ldots, n)$. It turns out that the x_j's are asymptotic to the zeros of P_n, which are the abscissae of the order n Gauss-Legendre quadrature rule. In numerical experiments, we found that for all n that we tried (from $n = 1$ up to $n = 200$ or so), using least squares to compute weights corresponding to the x_j's always resulted in positive weights, and that the subsequent Gauss-Newton iteration immediately achieved quadratic convergence.

Clearly, the choice of residual function plays a large role here; it is unclear whether a residual function exists which guarantees success for Algorithms 4.2 and 4.3.

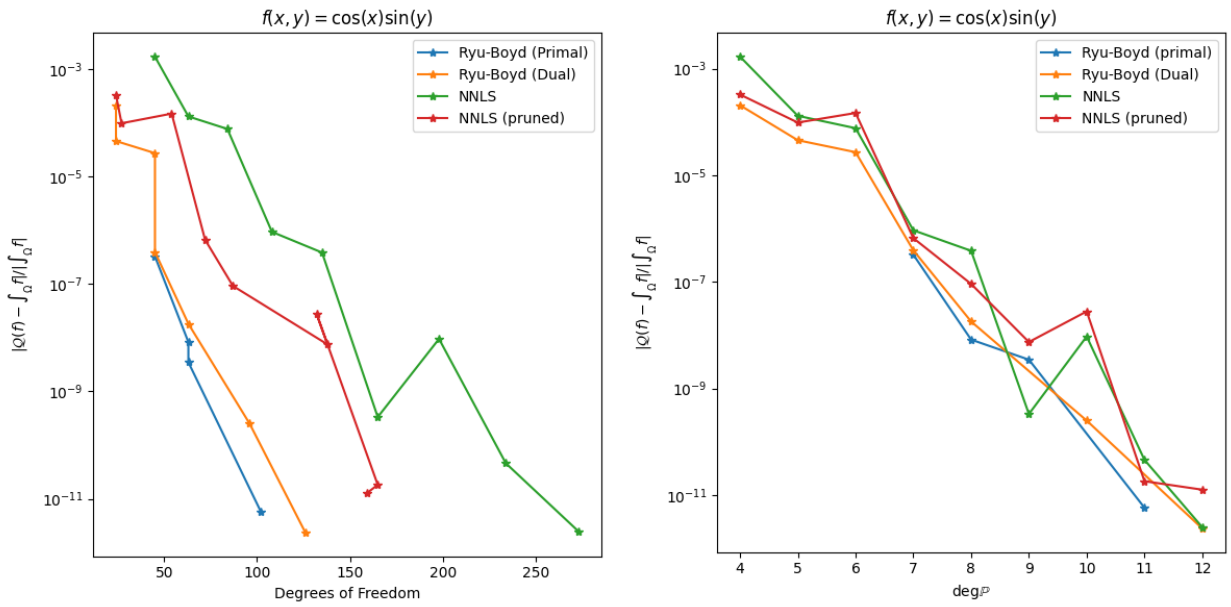

Figure 10: Relative integration error using each quadrature rule to integrate $f(x, y) = \cos(x)\sin(y)$ over Ω. Here, Ω is the cut cell with a disk-shaped bite removed, as shown in e.g. Figure 8. *Left*: plotted vs. $\mathrm{dof}(\mathcal{Q})$. *Right*: plotted vs. the degree of the polynomial space being integrated.

5 Numerical results

Each of these algorithms follows the same steps, at least conceptually.

1) Discretize the bounding box of Ω into a uniform grid and discard all points lying outside the domain.
2) Compute the moments of each basis function in the polynomial space being integrated.
3) Solve some version of the moment-matching equations using an algorithm described in the preceding sections.

To avoid overcomplicating things, we assume that Ω is described by a level set function and that we know its exact bounding box. In this case, Step 1 is extremely simple. But this need not be the case—if Ω is described by a CAD B-rep conforming to $\partial\Omega$, then computing the bounding box and subsequently filtering points outside Ω is more complicated. We do not dwell on the details here.

See Figures 10 to 12 for numerical results in 2D. In particular, we consider the domain:

$$\Omega = [-1, 1]^2 \setminus \left\{(x, y) \in \mathbb{R}^2 : (x-1)^2 + (y-1)^2 < 1\right\}. \tag{53}$$

In terms of the level set functions:

$$\psi_{\mathrm{box}}(x, y) = \max(\max(|x|, |y|) - 1 \tag{54}$$
$$\psi_{\mathrm{disk}}(x, y) = (x-1)^2 + (y-1)^2 - 1 \tag{55}$$
$$\psi_{\mathrm{diff}}(x, y) = \max(\psi_{\mathrm{box}}(x, y), -\psi_{\mathrm{disk}}(x, y)), \tag{56}$$

this can is equivalently given by:

$$\Omega = \left\{(x, y) \in \mathbb{R}^2 : \psi_{\mathrm{diff}}(x, y) \leqslant 0\right\}. \tag{57}$$

We use the monomial basis for a basis for $\mathbb{P}_{m,d}$:

$$\varphi_\alpha(x, y) = x^{\alpha_1} y^{\alpha_1} \qquad \text{s.t.} \qquad |\alpha| \leqslant m. \tag{58}$$

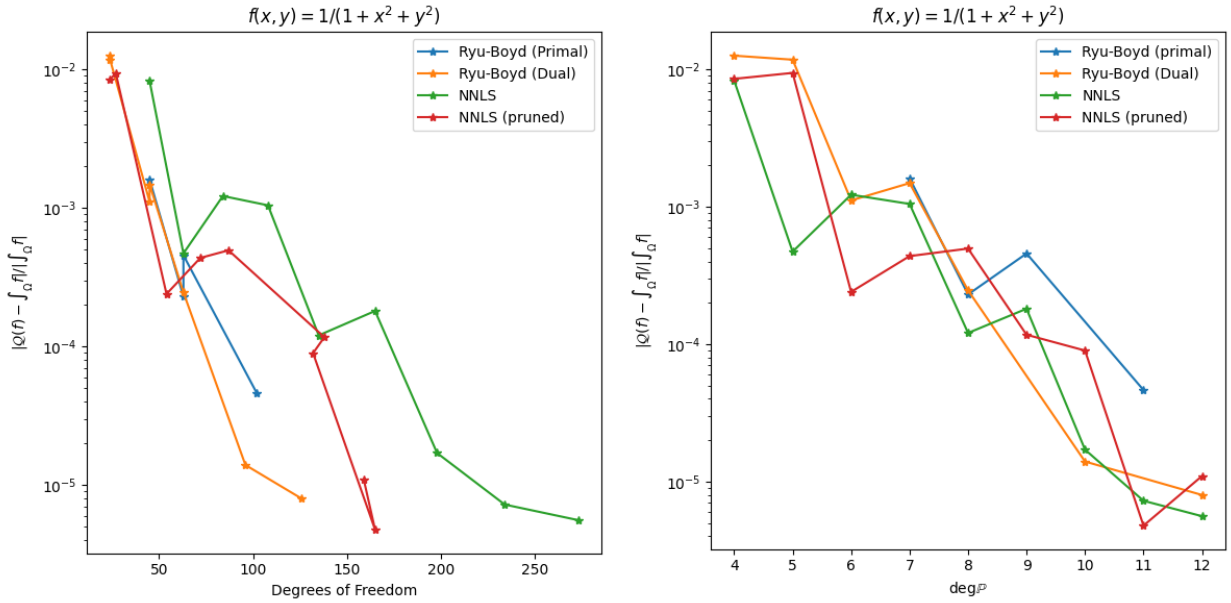

Figure 11: Relative integration error using each quadrature rule to integrate $f(x, y) = (1 + x^2 + y^2)^{-1}$ over Ω. *Left*: plotted vs. dof($\mathcal{Q}$). *Right*: plotted vs. the degree of the polynomial space being integrated.

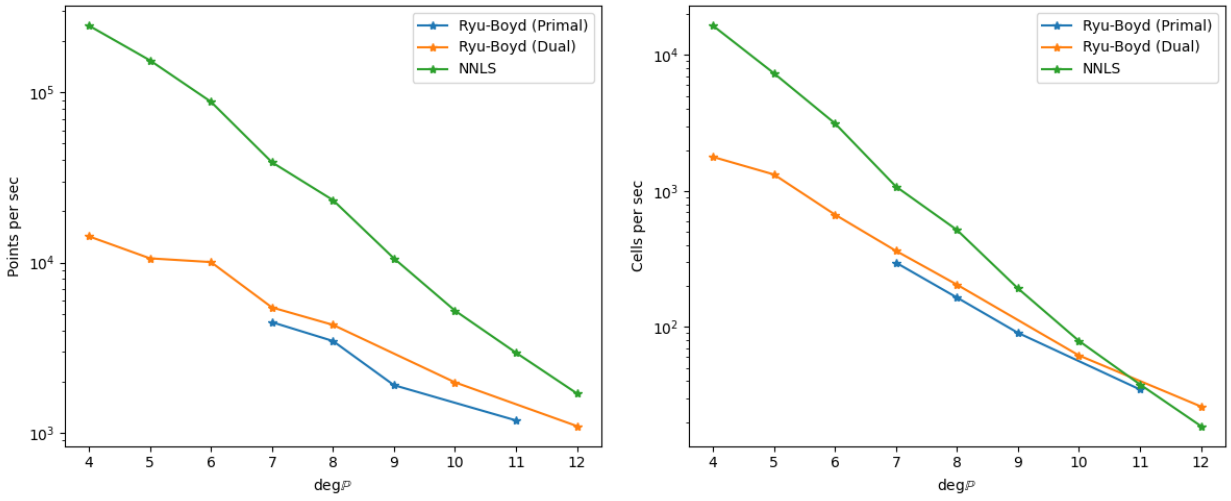

Figure 12: Timings result for building different quadrature rules on the cut cell shown in Figure 8. *Left*: numbers of quadrature points computed per second on a single core. *Right*: number of quadrature rules (i.e. cut cells processed) per second per core.

For small N (up to, say, $N = 15$ to 20), the monomial basis is accurate for our use, despite its inherent ill-conditioning. The target use case for the fast cut cell quadrature algorithms presented in this paper is numerical quadrature for the finite element method or similar, in which case N is unlikely to be excessively large. For higher degrees, the Chebyshev or Legendre bases could be used. Volume integration of multivariate monomials can be expedited by the divergence theorem. Specifically, it is not hard to show that:

$$\int_\Omega f(\boldsymbol{x})d\boldsymbol{x} = \frac{1}{d+q}\int_{\partial\Omega} \boldsymbol{n}(\boldsymbol{x})\cdot\boldsymbol{x}f(\boldsymbol{x})d\boldsymbol{x}. \tag{59}$$

One proof directly applies the divergence theorem; it can also be shown using Euler's theorem on homogeneous functions. If $\partial\Omega$ is polyhedral, we can continue to reduce this integral by repeatedly applying the divergence theorem within faces, then edges [6]. However, we argue that even in the polyhedral case, it is unlikely to be advantageous to further reduce the dimension of the boundary integral—the expressions become much more complicated and the gain in computational complexity is unclear.

6 Acknowledgments

Preliminary work for this research was carried out while employed at Coreform, LLC. Many thanks to the employees and management of Coreform for providing a fruitful and collaborative environment and for motivating this work in the first place. Special thanks are due to Drs. Derek Thomas, Michael Scott, Greg Vernon, David Kamensky, and Christopher Whetten for useful conversations and for providing helpful orientation within the world of computational mechanics.

References

[1] Atomic Decomposition by Basis Pursuit | SIAM Review. https://epubs.siam.org/doi/abs/10.1137/S003614450037906X.

[2] Alexander Barvinok. *A Course in Convexity*. Number 54 in Graduate Studies in Mathematics. American Mathematical Society, Providence, RI, 2002.

[3] Paul T. Boggs and Jon W. Tolle. Sequential Quadratic Programming. *Acta Numerica*, 4:1–51, January 1995.

[4] Hoang-Giang Bui, Dominik Schillinger, and Günther Meschke. Efficient cut-cell quadrature based on moment fitting for materially nonlinear analysis. *Computer Methods in Applied Mechanics and Engineering*, 366:113050, July 2020.

[5] Erik Burman, Susanne Claus, Peter Hansbo, Mats G. Larson, and André Massing. CutFEM: Discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*, 104(7):472–501, 2015.

[6] Eric B. Chin, Jean B. Lasserre, and N. Sukumar. Numerical integration of homogeneous functions on convex and nonconvex polygons and polyhedra. *Computational Mechanics*, 56(6):967–981, December 2015.

[7] Mathieu Collowald and Evelyne Hubert. Algorithms for computing cubatures based on moment theory. *Studies in Applied Mathematics*, 141(4):501–546, 2018.

[8] Philip J Davis. A Construction of Nonnegative Approximate Quadratures.

[9] Giacomo Elefante, Alvise Sommariva, and Marco Vianello. CQMC: An improved code for low-dimensional Compressed Quasi-MonteCarlo cubature.

[10] Wadhah Garhuom and Alexander Düster. Non-negative moment fitting quadrature for cut finite elements and cells undergoing large deformations. *Computational Mechanics*, 70(5):1059–1081, November 2022.

[11] Jan Glaubitz. Constructing Positive Interpolatory Cubature Formulas, September 2020.

[12] Jan Glaubitz. Stable high-order cubature formulas for experimental data. *Journal of Computational Physics*, 447:110693, December 2021.

[13] Jan Glaubitz. Construction and application of provable positive and exact cubature formulas. *IMA Journal of Numerical Analysis*, 43(3):1616–1652, June 2023.

[14] Ming Gu and Stanley C Eisenstat. Efficient Algorithms for Computing a Strong Rank-Revealing QR Factorization. *SIAM Journal on Numerical Analysis*, 17(4), 1996.

[15] Daan Huybrechs. Stable high-order quadrature rules with equidistant points. *Journal of Computational and Applied Mathematics*, 231(2):933–947, September 2009.

[16] John D. Jakeman and Akil Narayan. Generation and application of multivariate polynomial quadrature rules. *Computer Methods in Applied Mechanics and Engineering*, 338:134–161, August 2018.

[17] Vahid Keshavarzzadeh, Robert M. Kirby, and Akil Narayan. Numerical Integration in Multiple Dimensions with Designed Quadrature. *SIAM Journal on Scientific Computing*, 40(4):A2033–A2061, January 2018.

[18] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problem*. Number CL15 in Classics in Applied Mathematics. SIAM, 1995.

[19] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. On interpolation and integration in finite-dimensional spaces of bounded functions. *Communications in Applied Mathematics and Computational Science*, 1(1):133–142, May 2007.

[20] Sanjay Mehrotra. On the Implementation of a Primal-Dual Interior Point Method. *SIAM Journal on Optimization*, 2(4):575–601, November 1992.

[21] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, second edition edition, 2006.

[22] Ernest K. Ryu and Stephen P. Boyd. Extensions of Gauss Quadrature Via Linear Programming. *Foundations of Computational Mathematics*, 15(4):953–971, August 2015.

[23] R. I. Saye. High-Order Quadrature Methods for Implicitly Defined Surfaces and Volumes in Hyperrectangles. *SIAM Journal on Scientific Computing*, 37(2):A993–A1019, January 2015.

[24] Vaidyanathan Thiagarajan and Vadim Shapiro. Adaptively weighted numerical integration over arbitrary domains. *Computers & Mathematics with Applications*, 67(9):1682–1702, May 2014.

[25] B. Vioreanu and V. Rokhlin. Spectra of Multiplication Operators as a Numerical Tool. *SIAM Journal on Scientific Computing*, 36(1):A267–A288, January 2014.

[26] Stephen J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[27] Hong Xiao and Zydrunas Gimbutas. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. *Computers & Mathematics with Applications*, 59(2):663–676, January 2010.