Carathéodory cubature measures

Maria Tchernychova

St Anne's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy in Mathematics

Michaelmas 2015

To my readers

Abstract

We introduce an efficient algorithm for computing Carathéodory cubature measures, which are defined as interpolatory cubature measures with positive weights, whose cardinality grows polynomially with dimension, as proved in [16]. We discuss two Carathéodory cubature problem formulations. Both are based on thinning the support of the Cartesian product cubature measure, whose cardinality grows exponentially with dimension, via a formulation of a suitable feasibility LP (Linear Programming) problem. A basic feasible solution to the latter fully characterises a Carathéodory cubature measure.

The first problem formulation, initially presented in [48], employes the Simplex Algorithm or Interior Point Method to construct a basic feasible solution to the aforementioned LP problem. The complexity of this method is dependent on the number of nodes in the Cartesian product cubature and thus grows exponentially with dimension.

The second problem formulation constitutes the main contribution of the present work. Starting from the LP problem, arising from the Cartesian product cubature construction, we employ a hierarchical cluster representation of the underlying constraint matrix and the strictly feasible solution, arising from the weights of the Cartesian product cubature. Applying the Recombination Algorithm, introduced in [96], to this hierarchical data structure, we recursively generate a sequence of smaller LP problems. We construct a basic feasible solution to each LP problem in turn, by employing a novel algorithm, based on the SVD (Singular Value Decomposition) of the constraint matrix, culminating in a basic feasible solution for the original LP problem. The complexity of this algorithm, is independent of the number of nodes in the Cartesian product cubature, and can be shown to grow polynomially rather than exponentially with dimension. Moreover, the novel SVD-based method for computing basic feasible solutions, produces a one order of magnitude speed-up of the overall algorithm, when compared to the algorithm in [96], and is therefore preferable.

Acknowledgements

I am deeply indebted to Prof. Terry Lyons for being my supervisor and my mentor. Among my colleagues and fellow students, I owe many thanks to Dr. Lajos Gergely Gyurko for his helpful comments on my transfer thesis, Wei Pan and Youness Boutaib for our valuable discussions. Thanks are also extended to my confirmation thesis examiners Prof. Mike Giles and Prof. Christoph Reisinger for their detailed comments and suggestions. I thankfully acknowledge the financial support of the EPSRC and the Oxford-Man Institute of Quantitative Finance and I am also grateful to the Advanced Research Computing team for their support and assistance with High Performance Computing facilities.

Most importantly, I would like to thank my parents and my dear friend Ged Clarke for their support and generosity in all respects.

Contents

1 Background and Significance **1**

1.1 Univariate Quadrature Review 1

1.1.1 Newton-Cotes Formulae 5

1.1.2 Clenshaw-Curtis Formulae 6

1.1.3 Gaussian Formulae 7

1.1.3.1 Gauss-Legendre Quadrature 10

1.2 Multivariate cubature Review 12

1.2.1 Multivariate Cubature Methods 20

1.2.2 Monte Carlo Methods 21

1.2.2.1 Stratified Sampling 22

1.2.2.2 Importance Sampling 22

1.2.2.3 Control Variates 23

1.2.3 Quasi Monte Carlo 24

1.2.4 Smolyak Cubature 27

1.2.4.1 Clenshaw-Curtis Sparse Grids 30

1.2.4.2 Delayed Clenshaw-Curtis Sparse Grids 31

1.3 Contributions and Significance 34

1.3.1 Carathéodory Cubature Problem Formulation 34

1.3.2 Carathéodory Cubature with Simplex problem formulation . . . 38

1.3.2.1 Computational Complexity of the Carathéodory Cubature with Simplex problem formulation 45

1.3.3 Carathéodory cubature with SVD problem fomulation 46

1.3.3.1 Formulation of general framework 46

1.3.3.2 Hierarchical Clustering 48

1.3.3.3 SVD Algorithm for Basic Feasible Solution Construction 50

1.3.3.4 Recombination Algorithm 53

1.3.3.5 Computational Complexity of the Carathéodory Cubature with SVD problem formulation 61

1.3.4 Carathéodory cubature precision 63
1.4 Primal, Dual Simplex Algorithms and Interior Point Method 66
1.4.0.1 Primal Simplex Algorithm 66
1.4.0.2 Dual Simplex Algorithm 76
1.4.0.3 Primal-Dual Interior Point Method 78

2 Generalised Cubature Measures **81**

3 Recombination Algorithm **90**
3.1 Recursive Halving Forest Construction Algorithm 94
3.1.0.4 L-Tree forms 101
3.1.0.5 Tree Permutations 105
3.1.0.6 Reduced Measure 106
3.1.0.7 Updating Binary Tree Weights 108
3.1.0.8 Reduced Tree Measure 111
3.1.0.9 Tree Splitting 115
3.1.0.10 Construction of Reduced Tree Measures 120
3.2 1-Tree Measure Reduction Algorithm 122
3.2.0.11 Theoretical Complexity of the Recombination Algorithm with 1-Tree Measure Reduction 128
3.2.0.12 Updating the Recombination Algorithm with 1-Tree Measure Reduction 131
3.3 M-Tree Measure Reduction Algorithm 137
3.3.0.13 Theoretical Complexity of the Recombination Algorithm with M-Tree Measure Reduction 148
3.3.1 Degenerate Measure Reduction Algorithms and Practical Implementation Issues 151
3.3.2 Multiplicity of reduced tree measures 157
3.3.3 Multiplicity of Ψ-generalised Carathéodory cubature measures . 160
3.3.4 Stability of the M-Tree Measure Reduction Algorithm 161
3.3.4.1 Numerical ϵ-rank and Eigenspectrum of rank-deficient matrices 172

4 Carathéodory cubature measures **176**

5 Numerical Validation **189**
5.0.5 Introduction 189
5.0.6 Testing Procedures 191

5.0.7 Comparison of Carathéodory Cubature Algorithm Implementations with Dual Simplex and Interior Point Method and SVD Solver 209
5.0.8 Scalability of Carathéodory Cubature Algorithm with SVD Solver 223
5.1 Conclusions 225

Bibliography **227**

List of Figures

3.1 Rank-Deficient Degeneracy 152
3.2 Convex Hull Degeneracy 154
3.3 Eigenspectrum Figure 1 174
3.4 Eigenspectrum Figure 2 175

5.1 Oscillatory, Dim=2 197
5.2 Product Peak, Dim=2 197
5.3 Corner Peak, Dim=2 198
5.4 Gaussian, Dim=2 198
5.5 Continous, Dim=2 199
5.6 Discontinuous, Dim=2 199
5.7 Oscillatory, Dim=3 200
5.8 Product Peak, Dim=3 200
5.9 Corner Peak, Dim=3 201
5.10 Gaussian, Dim=3 201
5.11 Continous, Dim=3 202
5.12 Discontinuous, Dim=3 202
5.13 Oscillatory, Dim=4 203
5.14 Product Peak, Dim=4 203
5.15 Corner Peak, Dim=4 204
5.16 Gaussian, Dim=4 204
5.17 Continous, Dim=4 205
5.18 Discontinuous, Dim=4 205
5.19 Log-log plot of Wall-Clock ticks: Gurobi Dual Simplex, IBM CPLEX Dual Simplex, SVD 211
5.20 Log-log plot of Floating Point Operations: Gurobi Dual Simplex, IBM CPLEX Dual Simplex, SVD 212
5.21 Resident and Virtual Memory: Gurobi Dual Simplex, IBM CPLEX Dual Simplex, SVD 213

5.22 Oscillatory, Dim=4 215
5.23 Product Peak, Dim=4 215
5.24 Corner Peak, Dim=4 216
5.25 Gaussian, Dim=4 216
5.26 Continuous, Dim=4 217
5.27 Discontinuous, Dim=4 217
5.28 Numerical Errors Gurobi Barrier LP Solver 218
5.29 Numerical Errors IBM CPLEX Barrier LP Solver 219
5.30 Log-log plot of Wall-Clock ticks: Gurobi Barrier, IBM CPLEX Barrier, SVD 220
5.31 Log-log plot of Floating Point Operations: Gurobi Barrier, IBM CPLEX Barrier, SVD 221
5.32 Resident and Virtual Memory: Gurobi Barrier, IBM CPLEX Barrier, SVD 222
5.33 Threading Performance, Problem Size M=8568 224
5.34 Threading Performance, Problem Size M=11628 224

List of Tables

1.1 Smolyak Clenshaw-Curtis and Delayed Smolyak Clenshaw-Curtis . . . 33
1.2 Cardinality of full tensor Gaussian Cubature and Carathéodory Gaussian Cubature . . . 65

3.1 Eigenspectrum table 1 . . . 174
3.2 Eigenspectrum table 2 . . . 175

4.1 N_ℓ^* = Cardinality of Cartesian tensor Gaussian Cubature and $M_\ell^{'}$ = Cardinality of Carathéodory Gaussian Cubature . . . 186

5.1 Genz Function Suite Parameters . . . 192
5.2 Cardinalities of $Q^{CAR}(m,d)$ for dimensions $d = 2,3,4$, degree $m = 2,4,\ldots,20$ . . . 194
5.3 Cardinalities of $Q^{SM}(\ell,2)$ and $Q^{SMD}(\ell,2)$ in dimension $d = 2$ . . . 194
5.4 Cardinalities of $Q^{SM}(\ell,3)$ and $Q^{SMD}(\ell,3)$ in dimension $d = 3$ . . . 195
5.5 Cardinalities of $Q^{SM}(\ell,4)$ and $Q^{SMD}(\ell,4)$ in dimension $d = 4$ . . . 196
5.6 Negative Weights in $Q^{SM}(\ell,d)$ and $Q^{SMD}(\ell,d)$ in dimension $d = 2$, level ℓ . . . 207
5.7 Negative Weights in $Q^{SM}(\ell,d)$ and $Q^{SMD}(\ell,d)$ in dimension $d = 3$, level ℓ . . . 207
5.8 Negative Weights in $Q^{SM}(\ell,d)$ and $Q^{SMD}(\ell,d)$ in dimension $d = 4$, level ℓ . . . 208
5.9 Problem size: $M = \binom{m+d}{d}$ for degree $m = 4$, dimensions $d = 3,4,\ldots,30$ 210

Chapter 1

Background and Significance

1.1 Univariate Quadrature Review

In this section we review the most commonly used methods for numerical integration in one dimension over bounded intervals. Let f be a real integrable function over $[a, b]$. Computing the integral

$$I[f] := \int_a^b f(x)dx \tag{1.1}$$

explicitly may be difficult or even impossible. A natural approach to approximating (1.1) is to replace f with an easily integrable approximation f_n, for some $n \in \mathbb{N}$, for example $f_n \in \Pi_{n-1}$ the space of one-dimensional polynomials of degree $\leq n-1$.

In this respect, one may employ the unique interpolating Lagrange polynomial of f of degree $n-1$ over a set of n distinct nodes : $f_n(x) = \sum_{i=1}^n f(x_i)l_i(x)$, to obtain

$$I[f_n] = \int_a^b f_n(x)dx = \sum_{i=1}^n \Big(\int_a^b l_i(x)dx \Big) f(x_i) := \sum_{i=1}^n \omega_i f(x_i) \tag{1.2}$$

where the coefficients of the interpolating polynomial turn out to be linear combinations of f at n points $\{x_1, \ldots, x_n\}$ and the weights ω_i are obtained by integrating the Lagrange characteristic polynomial l_i, associated with the node x_i:

$$\omega_i := \int_a^b l_i(x)dx = \int_a^b \prod_{\substack{1 \leq m \leq n \\ m \neq i}} \frac{(x - x_m)}{(x - x_i)} dx \tag{1.3}$$

The right hand-side formula in (1.2), can also be derived by using a different approach to polynomial interpolation. Given the nodes $\{x_1, \cdots, x_n\}$, the weights $\{\omega_1, \cdots, \omega_n\}$

are to be chosen in such a way that $I[f_n]$ integrates exactly the elements of the basis $\mathcal{B}(\Pi_n) = \{b_1, \cdots, b_n\}$. For the standard monomial basis $\{1, x^1, \ldots, x^{n-1}\}$ the weights can be obtained by solving the following linear system of equations:

$$\begin{aligned} \omega_1 + \ldots + \omega_n &= \int_a^b dx \\ \omega_1 x_1 + \ldots + \omega_n x_n &= \int_a^b x dx \\ &\vdots \\ \omega_1 x_1^{n-1} + \ldots + \omega_n x_n^{n-1} &= \int_a^b x^{n-1} dx \end{aligned} \tag{1.4}$$

The coefficient matrix for this system is the Vandermonde matrix $V(x_1, \ldots, x_n)$, whose determinant is given by, see [1]

$$det\big(V(x_1, \ldots, x_n)\big) = \prod_{i=2}^{n} \prod_{j=1}^{i-1} (x_i - x_j) \tag{1.5}$$

Given, that the nodes $x_1, \ldots, x_n$ are distinct, $V(x_1, \ldots, x_n)$ is nonsingular and hence there is a uniquely determined set of weights $\omega_1, \ldots, \omega_n$ satisfying the moment matching equations.

Definition 1.1.1. *We call an integration formula $Q^n := \sum_{i=1}^n \omega_i \delta_{x_i}$ with prescribed nodes $\{x_1, \cdots, x_n\} \subset [a, b] \subset \mathbb{R}$ an interpolatory quadrature formula of degree $(n-1)$, if its weights $\{\omega_1, \cdots, \omega_n\}$ are given by* (1.3) *or equivalently if its weights satisfy the moment matching equations* (1.4)*.*

Once constructed Q^n, we may use it to approximate $I[f]$, that is

$$I[f] = \int_a^b f(x) dx \approx \sum_{i=1}^{n} \omega_i f(x_i) = Q^n[f]$$

if f is well-approximated by polynomials of degree up to $(n-1)$ on $[a, b]$.

We define the degree of precision of a quadrature formula Q^n as the maximum $m \in \mathbb{N}$, for which

$$I[f] = Q^n[f] \qquad \forall f \in \Pi_m$$

Theorem 1.1.2. *[Theorem 5.2.1 [3]] Any n-node quadrature formula Q^n with degree of precision $m \geq n-1$ is interpolatory.*

For any integrand f, the accuracy with which $Q^n[f]$ approximates the integral is clearly related to the degree of precision of Q^n and also to the accuracy with which f itself can be approximated by polynomials. The latter depends on the smoothness of f, to specify it, we define $\mathcal{C}^r(a,b)$ as the set of all r-times differentiable functions on $[a,b]$, whose rth derivative is continuous on $[a,b]$. Then, it can be shown that for $f \in \mathcal{C}^r(a,b)$ with $||f^{(r)}||_\infty \leq M(< \infty)$, all interpolatory quadrature formulae with positive weights, satisfy

$$\left|R^n[f]\right| = \left|I[f] - Q^n[f]\right| \leq \frac{c_r(b-a)^{r+1}||f^{(r)}||_\infty}{n^r}$$

where c_r depends on r only, see [90].This gives us the following asymptotic error bound

$$\left|R^n[f]\right| = O(n^{-r}) \tag{1.6}$$

In many applications, we need to integrate a function $f : [a,b] \to \mathbb{R}$, with respect to some weight function $\rho : [a,b] \to \mathbb{R}$. The weight functions are often taken to be non-negative functions, whose expression is known in closed form and the quadrature formula Q^n is usually constructed by approximating only the integrand of f

$$I[f] = \int_a^b f(x)\rho(x)dx \approx \sum_{i=1}^{n} \omega_i f(x_i) = Q^n[f]$$

The definition of a quadrature formula in definition (1.1.1) is for the common case of the weight function $\rho(x) = 1$.

In the following, we present a theorem, relating the error in a quadrature formula to the error with respect to the L_∞ norm of the best polynomial approximation to f, whose conclusion is crucial in proving the convergence of quadratures as $n \to \infty$.

Theorem 1.1.3. *[Theorem 5.2.2 [3]] Any interpolatory quadrature Q^n with a degree of accuracy m satisfies*

$$\Big|R^n[f]\Big| := \Big| \int_a^b f(x)\rho(x)dx - \sum_{i=1}^{n} \omega_i f(x_i)\Big| \leq \left(\int_a^b |\rho(x)|dx + \sum_{i=1}^{n} |\omega_i| \right) e_m^*(f) \quad (1.7)$$

where

$$e_m^*(f) := \inf\{\ ||P_m - f||_\infty \ :\ P_m \in \Pi_m\ \}$$

If all quadrature weights of Q^n are non-negative, then

$$\Big|R^n[f]\Big| := \Big| \int_a^b f(x)\rho(x)dx - \sum_{i=1}^{n} \omega_i f(x_i)\Big| \leq 2\big(\sum_{i=1}^{n} \omega_i\big)\ e_m^*(f) \quad (1.8)$$

where ρ denotes a non-negative integrable weight function

If the coefficients of $\big\{Q^n\big\}_{n\geq 1}$ of interpolatory quadratures are non-negative for all n, then the error bound (1.8) implies the convergence

$$Q^n[f] \to \int_a^b f(x)\rho(x)dx \qquad \text{as} \qquad n \to \infty$$

for all $f \in \mathcal{C}[a,b]$. The aforementioned convergence result is due to Weierstrass approximation Theorem, see [1], which guarantees that

$$e_m^*(f) \to 0 \qquad \text{as} \qquad m \to \infty$$

for all continuous f. In addition, the relationship

$$e_m^*(f) \leq e_{m-1}^*(f), \qquad \forall m \in \mathbb{N}$$

together with (1.8), implies that the error in the interpolatory quadrature formulae with non-negative weights can be expected to decrease as the degree of accuracy m of the underlying quadrature increases. Therefore, among all n-node interpolatory quadrature formulae, those with non-negative weights and maximum degree of accuracy are considered to be state of the art. These are called Gaussian quadrature formulae and are discussed in detail in the following section.

Interpolatory quadrature formulae are uniquely characterised by the choice of the nodes $x_1, \ldots, x_n$. Amongst the most widely employed interpolatory quadratures are the Newton-Cotes quadrature formulae: utilising equidistant nodes, Gaussian quadrature formulae: whose nodes are the roots of a certain class of orthogonal polynomials and Clenshaw-Curtis formulae: whose nodes are the roots or extrema of Chebyshev's polynomials.

1.1.1 Newton-Cotes Formulae

Newton-Cotes formulae are based on Lagrange interpolation with equally spaced nodes in $[a, b]$, for a fixed $n \geq 0$, the nodes are given by $x_k = x_0 + kh$, $k = 0, \ldots, n$. The midpoint, trapezoidal and Simpson formulae are special instances of Newton-Cotes formulae, taking $n = 0$, $n = 1$ and $n = 2$ respectively. Two types of Newton-Cotes formulae may be defined:

$$\begin{aligned} &\text{closed}: \qquad x_0 = a, x_n = b \quad \text{and} \quad h = \frac{b-a}{n} \\ &\text{open}: \qquad x_0 = a + h, x_n = b - h \quad \text{and} \quad h = \frac{b-a}{n+2} \end{aligned}$$

A significant property of the Newton-Cotes formulae is that while the weights of low order Newton-Cotes are all positive, all quadratures for $n \geq 14$ contain some negative weights, hence the relationship

$$\sum_{i=1}^{n} |\omega_i| > b - a$$

holds for Newton-Cotes of higher degree. In particular, the following theorem holds

Theorem 1.1.4 (Kusmin)**.** *Let $\omega_1^{(n)}, \ldots, \omega_n^{(n)}$ be the coefficients of closed Newton-Cotes formulae Q^n. Then*

$$\sum_{i=1}^{n} |\omega_i^{(n)}| \to \infty \qquad as \qquad n \to \infty \tag{1.9}$$

The increase in the norm of the weights with n signifies an amplified impact on the errors in the function values $f(x_i)$ and hence for large n, Newton-Cotes become numerically unstable. As noted in [95], the sum of the weights of any quadrature formula is the measure of the domain of integration, thus clearly if the weights are all positive, there is a natural bound to their size. If, however, some weights are negative, they can be very large relative to the domain of integration. When the weights are very large, some individual terms in the quadrature sum may be much larger than the sum itself. Hence, the quadrature formula, may lead to subtracting large numbers from other large numbers thereby getting a small result. Consequently this process can magnify any errors present in the individual terms, rendering the quadrature formula numerically unstable. For example, as noted in [15], $\sum |\omega_j| / \sum \omega_j \approx 10^{11}$ for

Newton-Cotes $n = 40$, which leads to a tremendous amplification of roundoff errors in the computation of $f(x_j)$.

The detrimental effects of negative weights are not limited to possible amplification of errors in the values of f, but also enter into the estimates of the error bounds for quadrature formulae and into the theory of their convergence properties. When proving the convergence properties of Newton-Cotes formulae for all $f \in \mathcal{C}[a, b]$, (1.7) and (1.8) do not yield useful bounds, as the sum of Newton-Cotes weights is unbounded. Indeed, it can be shown that Newton-Cotes formulae do not converge for all integrands which are analytic in the interval of integration, see [4].

Newton-Cotes formulae of low order are important building blocks for discretizations of differential equations and other numerical methods. Their practical significance stems from the fact that lower order Newton-Cotes formulae are useful in compound quadratures, obtained by domain decomposition and application of individual quadratures to each subdomain.

1.1.2 Clenshaw-Curtis Formulae

The Clenshaw-Curtis formulae use non-equidistant abscissas given as the zeros or the extreme points of the Chebyshev polynomials. The zeros of Chebyshev polynomials are given by

$$x_i = -\cos\left(\frac{\pi \cdot (2i-1)}{2n}\right) \qquad i = 1, \ldots, n$$

and the corresponding extrema are given by

$$x_i = -\cos\left(\frac{\pi \cdot (i-1)}{n-1}\right) \qquad i = 1, \ldots, n$$

The first set of nodes is a variant referred to as "classical" Clenshaw-Curtis or Fejer's first rule, see [5] and the second variant is known as "practical" Clenshaw-Curtis. The latter is generally considered to be more accurate, see [6] and also has the advantage of node reuse when n is doubled, therefore this variant is mainly employed in practice. The weights for the "practical" Clenshaw-Curtis quadratures are given by

$$\omega_1 = \omega_n = \frac{1}{n(n-2)} \qquad \text{and}$$

$$\omega_i = \frac{2}{n-1}\left(1 + 2 \cdot \sum_{j=1}^{(n-1)/2} \frac{1}{1-4j^2} \cdot cos\big(\frac{2\pi(i-1)j}{n_l^1 - 1}\big)\right) \qquad \text{for} \quad i = 2, \ldots, n-1$$

Clenshaw-Curtis quadrature formulae, based on the aforementioned nodes and weights approximate a given function more accurately, than interpolation polynomials based on equidistant abscissas, see [7] and hence are more suitable for numerical integration purposes than Newton-Cotes. For example, Clenshaw-Curtis formulae for the weight function $\rho(x) = 1$ have positive weights, see [5], [8], which guarantees the convergence

$$Q_{CC}^n[f] \to I[f] \qquad \text{as} \qquad n \to \infty, \qquad \forall f \in \mathcal{C}[a,b]$$

more generally , Clenshaw-Curtis formulae for weight functions

$$\rho \in L_p[a,b], \qquad p > 1$$

can be shown to converge for all Riemann integrable functions f, see [9]. The degree of accuracy of an n-node Clenshaw-Curtis quadrature formula is in general the same as Newton-Cotes, exact only up to degree $n-1$. However, in [10] experimental comparison for certain classes of functions, namely those that are not analytic in a large neighbourhood of $[-1,1]$, the speed of convergence as $n \to \infty$ of Clenshaw-Curtis and Gaussian quadrature formulae covered in the next section, whose degree of precision is $2n-1$, is found to be very similar. Thus, for such functions Clenshaw-Curtis essentially never requires many more function evaluations than the state of the art Gaussian quadrature formulae.

1.1.3 Gaussian Formulae

By Theorem 1.1.2 all interpolatory quadrature formulae have degree of accuracy $m \geq n-1$. If the quadrature nodes $\{x_1, \ldots, x_n\}$ are fixed in advance, one can generally hope to solve the moment matching system of m linear equations in n unknowns: the quadrature weights $\{\omega_1, \ldots, \omega_n\}$ only for $m \leq n-1$. However, Gauss showed that by adequately choosing the n nodes as well as the n weights, one can solve the moment matching equations for degree $m = 2n-1$, so as to obtain an n-point quadrature formula of degree $2n-1$. The following theorem, due to Jacobi, tells us the sufficient and necessary conditions for finding suitable nodes such that the degree of exactness m of an n-node interpolatory quadrature is greater than $n-1$.

Theorem 1.1.5. *[Theorem 10.1 [11]] The quadrature $Q^n[f] = \sum_{i=1}^n \omega_i f(x_i)$ has degree of exactness $m = (n-1) + l$, $l > 0$ iff it is interpolatory and the nodal polynomial $s_n(x) = (x - x_1)(x - x_2)\dots(x - x_n)$ is such that*

$$\int_a^b r(x)s_n(x)\rho(x)dx = 0 \qquad \forall r \in \Pi_{l-1} \tag{1.10}$$

where ρ denotes a non-negative weight function (e.g. $\rho(x) = 1$).

Corollary 1.1.6. *[Corollary 10.1 [11]] The maximum degree of exactness of the quadrature formula in Theorem 1.1.5 is $2n - 1$*

Setting $l = n$ and hence $m = 2n - 1$, the maximum admissible value in Theorem 1.1.5, we conclude that the nodal polynomial s_n satisfies

$$\int_a^b r(x)s_n(x)\rho(x)dx = 0, \qquad \forall r \in \Pi_{n-1} \tag{1.11}$$

that is to say $s_n \in \Pi_n$ is ρ-orthogonal to all polynomials in Π_{n-1}.

We recall that, for any non-negative weight function ρ two polynomials $P \neq 0$ and $Q \neq 0$ are said to be ρ-orthogonal if

$$\langle P, Q\rangle_\rho = \int_a^b P(x)Q(x)\rho(x)dx = 0$$

Now, let $\mathcal{P}^\rho := \left\{P_k^\rho \ : \ k = 0, 1, 2, \dots\right\}$ be a set of pairwise ρ-orthogonal polynomials of $deg(P_k^\rho) = k$ for $k = 0, 1, 2, \dots$. The set $\mathcal{P}^\rho$ is said to be orthogonal with respect to the weight function ρ. Such sets of orthogonal polynomials can be obtained for any weight function by applying the Gram-Schmidt orthogonalisation procedure to the set of monomials $\{1, x^1, x^2, \dots\}$, see [12] and the Gram-Schmidt orthogonalisation theorem tells us that the members of two sets $\mathcal{P}^\rho = \{P_k^\rho\}$, $\tilde{\mathcal{P}}^\rho = \{\tilde{P}_k^\rho\}$ differ only by multiplicative constants and therefore have the same zeros. The Gram-Schmidt orthogonalisation of monic polynomials $\left\{P_k^\rho \ : \ k = 0, 1, 2, \dots\right\}$ with respect to the inner product $\langle\cdot,\cdot\rangle_\rho$ satisfies a simple three-term recursive relationship, for proof see [13]

$$\begin{cases} P_{k+1}^\rho(x) = (x - \alpha_k)P_k^\rho(x) - \beta_k P_{k-1}^\rho(x), & k \geq 0 \\ P_{-1}^\rho(x) = 0, \quad P_0^\rho(x) = 1 \end{cases}$$

where

$$\alpha_k = \frac{\langle xP_k^\rho, P_k^\rho\rangle_\rho}{\langle P_k^\rho, P_k^\rho\rangle_\rho}, \quad \beta_{k+1} = \frac{\langle P_k^\rho, P_k^\rho\rangle_\rho}{\langle P_k^\rho, P_k^\rho\rangle_\rho} \quad k \geq 0$$

The above relationship is generally numerically stable and is employed in the numerical computation of the roots of the underlying system of orthogonal polynomials, associated with a particular weight function ρ.

The most frequently occurring weight functions ρ and the underlying domains of integration are listed below:

Domain	Weight Function $\rho(x)$	System of Orthogonal Polynomials
$[-1,1]$	1	Legendre
$[-1,1]$	$(1-x^2)^{-1/2}$	Chebyshev, first kind
$[-1,1]$	$(1-x^2)^{1/2}$	Chebyshev, second kind
$[-1,1]$	$(1-x)^{\alpha}(1+x)^{\beta}$	Jacobi
$[0,\infty)$	e^{-x}	Laguerre
$[0,\infty)$	$x^{\alpha}e^{-x}$, $\alpha > -1$	Generalised Laguerre
$(-\infty,\infty)$	e^{-x^2}	Hermite

Returning to the result (1.11), it shows that the nodal polynomial s_n is equal up to a multiplicative constant, to the ρ-orthogonal polynomial P_n^{ρ} and its nodes $\{x_1,\ldots,x_n\}$ are the zeros of P_n^{ρ}. In order to be usable as quadrature nodes, the zeros $\{x_1,\ldots,x_n\}$ of P_n^{ρ} have to be simple, real and located inside the interval $[a,b]$. These requirements are indeed met, as is illustrated in [2].

Hence, the quadrature formula $Q_G^n = \sum_{i=1}^n \omega_i f(x_i)$, called Gaussian quadrature with abscissae $\{x_1,\ldots,x_n\}$ given by the zeros of P_n^{ρ}, called Gauss nodes and weights $\{\omega_1,\ldots,\omega_n\}$ given by

$$\omega_i = \int_a^b l_i(x)\rho(x)dx$$

where $l_i \in \Pi_n$ is the i-th characteristic Lagrange polynomial. Q_G^n has degree of exactness $2n-1$, the maximum value achieved by any interpolatory quadrature with n-nodes.

Gaussian quadrature formulae attain the maximum degree of accuracy for any admissible weight function ρ, irrespective of whether the domain of integration is bounded or not. The following convergence result, however only holds for bounded domains

of integration $[a, b]$. Since all Gaussian quadrature formulae have positive weights, by Theorem 5.24 in [3], the convergence

$$Q_G^n[f] \to I[f] \qquad \text{as} \quad n \to \infty \qquad \forall f \in \mathcal{C}[a, b]$$

can be derived by Theorem 1.1.3.

In the following we introduce the Gauss-Legendre family of orthogonal polynomials, which will be the constituent quadrature formulae of the multivariate cubature formulae, we will present in the subsequent sections.

1.1.3.1 Gauss-Legendre Quadrature

The Legendre polynomials are orthogonal polynomials over the interval $(-1, 1)$ with respect to the weight function $\rho(x) = 1$, defined as

$$L_n(x) = \frac{1}{2^n} \sum_{\ell=0}^{\lfloor n/2 \rfloor} (-1)^\ell \binom{n}{\ell} \binom{2n - 2\ell}{n} x^{n-2\ell} \qquad n = 0, 1, \dots$$

or recursively defined through the three-term relationship

$$\begin{cases} L_{n+1}(x) = \frac{2n+1}{k+1} L_n(x) - \frac{n}{n+1} L_{n-1}(x), \quad n \geq 0 \\ L_0(x) = 1, \quad L_1(x) = x \end{cases}$$

The Gauss-Legendre quadrature nodes x_j are given by the zeros of $L_n(x)$ and the corresponding weights are given by

$$\omega_j = \frac{2}{(1 - x_j^2)[L_n'(x_j)]^2}$$

We conclude this section by summarising that Newton-Cotes quadratures are impractical for highly smooth integrands $f \in \mathcal{C}^r[a, b]$, r large. For such integrands one should utilise high-order quadrature formulae in order to exploit all of the underlying integrand's smoothness. As noted earlier, high-order Newton-Cotes quadrature formulae are numerically unstable, due to the presence of negative weights, and thus should be avoided. Whilst for the Clenshaw-Curtis and Gaussian quadratures we have the following theorem

Theorem 1.1.7. *[Theorem 4.1 [10]] If* $Q^n = \sum_{i=1}^n \omega_i f(x_i)$ *is an interpolatory quadrature with non-negative weights, as is true for both Clenshaw-Curtis and Gaussian quadratures with standard weight function* $\rho(x) = 1$*, then for any* $f \in \mathcal{C}[a,b]$*, we have*

$$\Big|R^n[f]\Big| = \Big|I[f] - Q^n[f]\Big| \leq 2\big(\sum_{i=1}^n \omega_i\big)\ e^*_{n-1}(f) \tag{1.12}$$

and in the special case of Gausssian quadrature (1.12) *can be improved upon to*

$$\Big|R^n[f]\Big| = \Big|I[f] - Q^n_G[f]\Big| \leq 2\big(\sum_{i=1}^n \omega_i\big)\ e^*_{2n-1}(f) \tag{1.13}$$

where

$$e^*_m(f) := \inf\{\ ||P_m - f||_\infty\ :\ P_m \in \Pi_m\ \}$$

We note that $||P_m - f||_\infty = O(m^{-r})$ as $m \to \infty$ for $f \in \mathcal{C}^r[a,b]$, as noted in [15]. Thus, for highly smooth integrands f, utilising the highest in order of precision quadrature formula, i.e Gaussian quadrature, is clearly optimal in terms of speed of convergence.

1.2 Multivariate cubature Review

Unlike univariate quadrature formulae, most of which are based on polynomial interpolation, the construction of multivariate cubature formulae is intrinsically more difficult and the formulae produced do not generally rely on polynomial interpolation.

Firstly, we note that in one dimension it is sufficient to develop quadrature formulae for the three intervals $[0,1]$, $[0,\infty)$ and $(-\infty,\infty)$, as noted in [95], since the affine transformations mapping any interval into one of these intervals leave most quadrature properties, such as the degree of precision, unchanged. In two or more dimensions, however, there are infinitely many regions that are not equivalent under affine transformations. The usual approach to deal with this has been to develop formulae for the simplest regions: d-dimensional unit cube, simplex, sphere, etc. In the following, we will only consider multidimensional integration formulae over d-dimensional cubes.

The most intuitive approach to integrating a function f over a d-dimensional unit cube $[0,1]^d$ is by constructing the Cartesian product of univariate quadrature formulae on $[0,1]$. For this we recall, the following result

Definition 1.2.1. *(d-Fold Cartesian Product Rule) Suppose the integration region B is the Cartesian product of $d \geq 2$ regions $B_1, \ldots, B_d$, that is $B = B_1 \times \ldots \times B_d$. Let*

$$Q_{n_k}^{(k)}[f_k] = \sum_{j_k=1}^{n_k} \omega_{j_k}^k f_k(x_{j_k}^k) \qquad for \quad k \in [\![1,d]\!]$$

denote quadrature formulae for integrals

$$I_k[f_k] := \int_{B_k} f_k(x^k)dx^k \qquad for \quad k \in [\![1,d]\!]$$

Then, the d-fold Cartesian product formula $\left(Q_{n_1}^{(1)} \otimes \ldots \otimes Q_{n_d}^{(d)}\right)$ for the integral over B is given by

$$\left(Q_{n_1}^{(1)} \otimes \ldots \otimes Q_{n_d}^{(d)}\right)[f] := \sum_{j_1=1}^{n_1} \ldots \sum_{j_d=1}^{n_d} \omega_{j_1}^1 \ldots \omega_{j_d}^d f(x_{j_1}^1, \ldots, x_{j_d}^d)$$

where $[\![1,d]\!] := 1, \ldots, d$.

The following theorem relates the degree of exactness of the constituent univariate quadrature formulae to the d-fold Cartesian product formula.

Theorem 1.2.2. *[Theorem 6.1.2 [2]] If $Q_{n_k}^{(k)}$ integrates f_k exactly over B_k for $k \in [\![1, d]\!]$, then the Cartesian product rule $\left(Q_{n_1}^{(1)} \otimes \ldots \otimes Q_{n_d}^{(d)}\right)$ integrates the product*

$$f(x^1, \ldots, x^d) = f_1(x^1) \ldots f_d(x^d), \qquad \text{for} \quad x^k \in B_k, \quad k \in [\![1, d]\!] \tag{1.14}$$

exactly over $B = B_1 \times \ldots \times B_d$.

Hence if degree of exactness of $deg(Q_{n_k}^{(k)}) = m_k$, then we have the following

$$deg\Big(\left(Q_{n_1}^{(1)} \otimes \ldots \otimes Q_{n_d}^{(d)}\right)\Big) = \min\{m_1, \ldots, m_k\}$$

Clearly the number of abscissas in $\left(Q_{n_1}^{(1)} \otimes \ldots \otimes Q_{n_d}^{(d)}\right)$ is given by $N = n_1 \times \ldots \times n_d$.

Now let

$$I_d[f] = \int_{[0,1]^d} f(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}}, \qquad \underline{\boldsymbol{x}} = (x^1, \ldots, x^d) \tag{1.15}$$

If we let Q_d^n denote the Cartesian product of identical univariate quadratures of degree $m \in \mathbb{N}$, given by

$$Q_{n_1}^{(1)} = \ldots = Q_{n_d}^{(d)} = Q$$

then Q_d^n has $N = n^d$ nodes and $deg(Q_d^n) = m$ and Q_d^n) is given by

$$Q_d^n[f] := \big(Q \otimes Q \otimes \ldots \otimes Q\big)[f] = \sum_{j_1=1}^{n} \ldots \sum_{j_d=1}^{n} \omega_{j_1}^1 \ldots \omega_{j_d}^d f(x_{j_1}^1, \ldots, x_{j_d}^d) \tag{1.16}$$

As detailed in [95], if the integrand f satisfies

$$\left| \frac{\partial^m f}{(\partial x^i)^m} \right| \leq M \qquad \text{for} \quad i \in [\![1, d]\!] \tag{1.17}$$

for some constant M and integer m, so that, in each variable f is C^m uniformly with respect to the other variables, then there exists a constant K such that

$$\left| \int_{[0,1]} f(x^1, \ldots, x^d) dx^i - Q[f, x^i] \right| \leq K n^{-m} \qquad \text{for} \quad i \in [\![1, d]\!] \tag{1.18}$$

for all values of $x^1, x^2, \ldots, x^{i-1}, x^{i+1}, \ldots, x^d$ lying between 0 and 1. The estimate (1.17) holds similarly to (1.6) for the error of univariate quadrature with an underlying integrand $f : [a, b] \to \mathbb{R}$ satisfying $\left| \frac{\partial^m f}{(\partial x)^m} \right| \leq M$.

Firstly, we provide an error estimate for the two-dimensional Cartesian product formulae on the square $[0,1]^2$, for functions satisfying (1.17) and thus (1.18).

$$\begin{aligned}
&\Big| \int_0^1 f(x^1,x^2)dx^1dx^2 - \sum_{j_1=1}^{n}\sum_{j_2=1}^{n}\omega_{j_1}^1\omega_{j_2}^2 f(x_{j_1}^1,x_{j_2}^2)\Big| \\
&= \Big| \int_0^1 f(x^1,x^2)dx^1dx^2 - \sum_{j_1=1}^{n} f(x_{j_1}^1,x^2)dx^2 + \sum_{j_1=1}^{n}\omega_{j_1}^1 f(x_{j_1}^1,x^2)dx^2 \\
&- \sum_{j_1=1}^{n}\omega_{j_1}^1\sum_{j_2=1}^{n}\omega_{j_1}^1\omega_{j_2}^2 f(x_{j_1}^1,x_{j_2}^2)\Big| \\
&\leq \int_0^1 \Big| \int_0^1 f(x^1,x^2)dx^1 - \sum_{j_1=1}^{n}\omega_{j_1}^1 f(x_{j_1}^1,x^2)\Big|dx^2 \\
&+ \sum_{j_1=1}^{n}|\omega_{j_1}^1|\Big| \int_0^1 f(x_{j_1}^1,x^2)dx^2 - \sum_{j_2=1}^{n}\omega_{j_1}^1 f(x_{j_1}^1,x_{j_2}^2)\Big| \\
&\leq Kn^{-m} + \sum_{j_1=1}^{n}|\omega_{j_1}^1|Kn^{-m}
\end{aligned}$$

where the last inequality holds by applying (1.18) in both variables x^1 and x^2.

The extension of this result to higher dimensions is very similar. In general for dimension d, setting $A = \sum_{j=1}^{n}|\omega_j^i|$ for $i \in [\![1,d]\!]$, we obtain the following error estimate

$$\begin{aligned}
&\big|R_d[f]\big| = \big|I_d[f] - Q_d^n[f]\big| \\
&\leq Kn^{-m} + AKn^{-m} + A^2Kn^{-m} + \ldots + A^{d-1}Kn^{-m} \\
&= (1 + A + \ldots + A^{d-1})Kn^{-m} = Cn^{-m} = C(n^d)^{-\frac{m}{d}} = CN^{-\frac{m}{d}}
\end{aligned}$$

We note that the Cartesian product of univariate quadratures Q_d^n, derived in (1.16), satisfies the definition of a cubature formula of degree $m \in \mathbb{N}$, to be defined shortly hereafter. Additionally, we remark that whilst the bound on Q_d^n is only an upper bound on the error. N.S. Bahvalov, studied the question of lower error bounds for cubature formulae in d-dimensions. Indeed in [56], Bahvalov has shown that this bound cannot be much improved upon, by showing that for the family of functions, defined by condition (1.17), there is a constant $K_1 = K_1(m,M)$, such that for any N-node d-dimensional cubature formula Q_d^n there exists a function f in the family for which

$$\Big| \int_{[0,1]^d} f - Q_d^n[f]\Big| > K_1 N^{-\frac{m}{d}} \tag{1.19}$$

Hence, the error bound on the Cartesian product of univariate quadratures $Q_d^n[f]$, is given by

$$\Big|R_d^n[f]\Big| = \Big|I_d[f] - Q_d^n[f]\Big| = O(N^{-\frac{m}{d}}) \tag{1.20}$$

Therefore, for a fixed regularity m of the underlying integrand f, the complexity of numerical integration via Cartesian product of univariate quadratures grows exponentially with dimension d. This exponential explosion of function evaluations with dimension is commonly referred to as the "curse of dimensionality" or intractability. To overcome this, similarly to the univariate case we would like to derive interpolatory cubature formulae in dimension $d \geq 2$.

We recall that, univariate interpolatory quadrature formulae Q^n are obtained by integrating exactly interpolating polynomials or equivalently by requiring Q^n to be exact for all polynomials $P \in \Pi_{n-1}$. The following illustrates that the second condition is more suitable than the first for defining multivariate interpolatory cubature formulae.

Let $\Pi_m^d = \mathbb{R}_m[X_1, \ldots, X_d]$ denote the space of d-variate polynomials of degree up to m, then $dim(\Pi_m^d) = \binom{m+d}{m}$. For dimensions $d \geq 2$, the multidimensional Weierstrass Theorem shows that continuous functions f can be approximated with arbitrarily high accuracy by polynomials $P \in \Pi_m^d$ and a similar conclusion can be drawn from the multivariate Taylor expansion of sufficiently smooth functions. However, for $d \geq 2$, the dimensions of Π_m^d, $dim(\Pi_m^d)$ do not form a contiguous sequence of natural numbers

$$\left\{ dim(\Pi_m^d) \ : \ d \in \mathbb{N}_0 \right\} \neq \mathbb{N}$$

As noted in [2], the interpolation problem for prescribed function values $f(\underline{\boldsymbol{x}}_1), \ldots, f(\underline{\boldsymbol{x}}_n)$ can be solved if and only if the linear functionals

$$\underline{\boldsymbol{x}}_1^*, \ldots, \underline{\boldsymbol{x}}_n^* : \Pi_m^d \to \mathbb{R} \qquad \text{defined by} \quad \underline{\boldsymbol{x}}_i^*(P) = P(\underline{\boldsymbol{x}}_i) \quad i = 1, \ldots, n$$

are linearly independent over Π_m^d. If this independence requirement is met, the interpolating polynomial is unique if and only if $dim(\Pi_m^d) = n$. Firstly, the number of interpolation points n can only be chosen such that

$$n \in \left\{ dim(\Pi_m^d) \ : \ d \in \mathbb{N}_0 \right\}$$

and then, even if n is chosen such that $n = dim(\Pi_m^d)$, it may still happen that linear functionals $\underline{\boldsymbol{x}}_1^*, \ldots, \underline{\boldsymbol{x}}_n^*$ are not linearly independent over Π_m^d. The following example illustrates one such case of non-existence and non-uniqueness of interpolating polynomial in 2 dimensions.

Consider the interpolation of a function $f : \mathbb{R}^2 \to \mathbb{R}$ at the points $\underline{\boldsymbol{x}}_1 = [0, 0]^T$, $\underline{\boldsymbol{x}}_2 = [1/2, 1/2]^T$, $\underline{\boldsymbol{x}}_3 = [1, 1]^T$ using $P \in \Pi_1^2 = \{1, x, y\}$. Since the interpolation points

lie on the straight line $y = x$ in $\mathbb{R}^3$, all polynomials of the form $P = \lambda(y - x) \in \Pi_1^2$, $\lambda \in \mathbb{R}$ satisfy the interpolation conditions and hence this problem is generally unsolvable, even though $n = dim(\Pi_1^2)$.

Since the straightforward interpolation approach to multivariate cubature construction is not feasible, it is reasonable to construct cubature formulae by requiring them to integrate all d-variate polynomials of degree m exactly. This amounts to the following definition.

Definition 1.2.3. *A formula Q_d^n with prescribed nodes $\{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_n\} \subset \mathbb{R}^d$, is a cubature formula of degree $m \in \mathbb{N}$, if its weights $\{\omega_1, \ldots, \omega_n\}$, satisfy the following system of moment matching equations.*

$$\begin{aligned} \omega_1\gamma_1(\underline{\boldsymbol{x}}_1) + \ldots + \omega_n\gamma_1(\underline{\boldsymbol{x}}_1) &= \int \gamma_1(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}} \\ \omega_1\gamma_2(\underline{\boldsymbol{x}}_1) + \ldots + \omega_n\gamma_2(\underline{\boldsymbol{x}}_n) &= \int \gamma_2(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}} \\ &\vdots \\ \omega_1\gamma_M(\underline{\boldsymbol{x}}_1) + \ldots + \omega_n\gamma_M(\underline{\boldsymbol{x}}_n) &= \int \gamma_M(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}} \end{aligned} \tag{1.21}$$

where $M = dim(\Pi_m^d) = \binom{m+d}{d}$ and $\mathcal{B}(\Pi_m^d) = \{\gamma_1, \ldots, \gamma_M\}$, $\gamma_1(\underline{\boldsymbol{x}}) = 1$ is some fixed basis for Π_m^d.

Definition 1.2.4. *[Definition 6.2.3 [2]] A cubature formula Q_d^n with prescribed nodes $\{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_n\} \subset \mathbb{R}^d$ is said to be an interpolatory cubature of degree $m \in \mathbb{N}$ if its weights $\{\omega_1, \ldots, \omega_n\}$ are the unique solution of the system of the moment matching equations* (1.21).

The requirement that the weights $\{\omega_1, \ldots, \omega_n\}$ must be the unique solution of the moment matching equations in the definition of an interpolatory cubature, can be explained as follows. Let

$$\underline{\boldsymbol{x}}_1^*, \ldots, \underline{\boldsymbol{x}}_n^* : \Pi_m^d \to \mathbb{R} \qquad \text{defined by} \quad \underline{\boldsymbol{x}}_i^*(P) = P(\underline{\boldsymbol{x}}_i) \quad i = 1, \ldots, n$$

be the linear functionals pertaining to the nodes $\{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_n\}$. Using this notation, the moment matching equations in (1.2.3) can be written as follows

$$\sum_{i=1}^{n} \omega_i \underline{\boldsymbol{x}}_i^*(\gamma_j) = I[\gamma_j] \qquad j = 1, \ldots, M$$

The above equations can be solved if and only if the linear functional I over Π_m^d is a linear combination of functionals $\{\underline{\boldsymbol{x}}_1^*, \ldots, \underline{\boldsymbol{x}}_n^*\}$, that is, if and only if $I \in \text{span}\{\underline{\boldsymbol{x}}_1^*, \ldots, \underline{\boldsymbol{x}}_n^*\}$. If this is the case, the weights $\{\omega_1, \ldots, \omega_n\}$ of Q^n are chosen as the coefficients of the linear combination

$$I = \sum_{i=1}^{n} \omega_i \underline{\boldsymbol{x}}_i^* \qquad \text{over} \quad \Pi_m^d$$

This shows that the weights are uniquely determined by the nodes if and only if the functionals $\{\underline{\boldsymbol{x}}_1^*, \ldots, \underline{\boldsymbol{x}}_n^*\}$ are linearly independent over Π_m^d. If, for instance $\underline{\boldsymbol{x}}_n^*$ can be represented as a linear combination of the remaining functionals

$$\underline{\boldsymbol{x}}_n^* = \sum_{i=1}^{n-1} \beta_i \underline{\boldsymbol{x}}_i^* \qquad \text{over} \quad \Pi_m^d$$

then

$$I = \sum_{i=1}^{n-1} \omega_i \underline{\boldsymbol{x}}_i^* + \omega_n \sum_{i=1}^{n-1} \beta_i \underline{\boldsymbol{x}}_i^* = \sum_{i=1}^{n-1} (\omega_i + \omega_n \beta_i) \underline{\boldsymbol{x}}_i^* \qquad \text{over} \quad \Pi_m^d$$

and hence the solution $\{\omega_1, \ldots, \omega_n\}$ to the moment matching equations is not unique and therefore Q^n is not interpolatory. Since Q^n is not interpolatory, there exists an interpolatory cubature formula Q^{n-1}, with $(n-1)$ nodes which satisfies the moment matching equations. We can continue this argument, decreasing the number of nodes in the cubature formula, until we find an interpolatory cubature. Thus for any non-interpolatory cubature formula Q^n, there is an interpolatory cubature formula $Q^{n'}$, with $n' < n$, which satisfies the same moment matching equations and since the number of nodes to attain a given accuracy should be kept as low as possible $Q^{n'}$ is preferable to Q^n.

Whenever Q^n is interpolatory, the functionals $\{\underline{\boldsymbol{x}}_1^*, \ldots, \underline{\boldsymbol{x}}_n^*\}$ associated with the nodes of Q^n are linearly independent. Since the space of all linear functionals over Π_m^d, has the same dimension as Π_m^d, $dim(\Pi_m^d)$ is the maximum number of linearly independent functionals over Π_m^d and hence an upper bound on the cardinality of Q^n is established, $n \leq dim(\Pi_m^d) = \binom{m+d}{m}$.

We note that the system of moment matching equations (1.21) consists of $\binom{m+d}{d}$ distinct equations in $n(d+1)$ variables, n weights and nd coordinates of the nodes. So if

n is greater than or equal to

$$\left\lfloor \frac{1}{(d+1)} \binom{m+d}{d} \right\rfloor + 1 \tag{1.22}$$

the number of variables will be at least as great as the number of equations. However, as noted in [95], since the resulting system is non-linear, we do not if a solution exists whenever n is greater than (1.22), but we do know cases of solutions which exist for much smaller n.

If on the other hand, we fix the nodes $\{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_n\}$ in advance, the system of equations (1.21) becomes linear in only n variables $\{\omega_1, \ldots, \omega_n\}$. The nodes $\{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_n\}$ can always be chosen as the nodes arising from the tensor product of d univariate quadratures of degree $m \in \mathbb{N}$. For m, d sufficiently large the cardinality n of nodes arising from the Cartesian product, satisfies $n \geq \binom{m+d}{d}$, then the system of moment matching equations in (1.21) certainly has a solution. However, the cubature formula, resulting from the product of univariate quadratures will not be interpolatory, since $n \geq \binom{m+d}{d}$. Indeed, for a cubature formula to be interpolatory, we have the following bounds on the cardinality n of its nodes.

Theorem 1.2.5. *[Theorem 6.2.3, [2]] The number n of nodes in any interpolatory cubature Q^n with degree of exactness m satisfies*

$$\binom{\lfloor \frac{m}{2} \rfloor + d}{d} = dim(\Pi^d_{\lfloor \frac{m}{2} \rfloor}) \leq n \leq dim(\Pi^d_m) = \binom{m+d}{d} \tag{1.23}$$

Hence, to construct an n-node interpolatory cubature formula of degree m, we must take n to be equal to at least the lower bound (1.23), and it is sufficient to take n equal to the upper bound. There is no assurance that cubature formulae that achieve the lower bound in (1.23) can be found, such cubatures have only been found for low dimensions and small degrees of accuracy.

It is widely asserted, see [95], [2] that three desirable properties of any d-dimensional

cubature formula of degree $m \in \mathbb{N}$ are as follows:

1) Number of nodes n should be as small as possible, $n \leq \binom{m+d}{d}$ (1.24)

2)Nodes should be located inside the region of integration

3)Weights should be non-negative

Tchakaloff's Theorem [16] asserts the existence of cubature formulae of any degree $m \in \mathbb{N}$ in d dimensions, satisfying the three aforementioned properties, with respect to a positive compactly supported measure, which is absolutely continuous with respect to the d-dimensional Lebesgue measure. The resulting cubature formulae are interpolatory. We will provide a detailed discussion of Tchakaloff's Theorem and its extensions to general Borel sets $R \subseteq \mathbb{R}^d$ for positive Borel measures on $\mathbb{R}^d$ in Chapter 2.

Completely analogously to the univariate quadrature formulae, the following theorem insures the convergence of interpolatory cubature formulae with non-negative weights.

Theorem 1.2.6. *[Theorem 6.2.4 [2]] Any interpolatory cubature formula Q^n with a degree of accuracy m satisfies*

$$\left|R^n[f]\right| := \left| \int_R f(\underline{\boldsymbol{x}})\rho(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}} - \sum_{i=1}^{n} \omega_i f(\underline{\boldsymbol{x}}_i)\right| \leq \left(\int_R \rho(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}} + \sum_{i=1}^{n} |\omega_i| \right)\ e^*_m(f) \quad (1.25)$$

where

$$e^*_m(f) := \inf\{\ ||P_m - f||_\infty\ :\ P_m \in \Pi^d_m\ \}$$

If all cubature weights are non-negative, then

$$\left|R^n[f]\right| := \left| \int_R f(\underline{\boldsymbol{x}})\rho(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}} - \sum_{i=1}^{n} \omega_i f(\underline{\boldsymbol{x}}_i)\right| \leq 2\big(\sum_{i=1}^{n} \omega_i\big)\ e^*_m(f) \quad (1.26)$$

where ρ denotes a non-negative integrable weight function

If the weights of interpolatory cubatures $\left\{Q^n_m\right\}_{n\geq 1}$ of degree m are non-negative for all n, then the error bound implies convergence

$$Q^n_m[f] \to \int_R f(\underline{\boldsymbol{x}})\rho(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}} \qquad \text{as} \qquad m \to \infty$$

for all $f \in \mathcal{C}(R)$, provided the integration region R is bounded and closed. Similarly to the univariate quadrature formulae, the error in cubature formulae with non-negative weights decreases as the degree of accuracy increases.

1.2.1 Multivariate Cubature Methods

About four decades ago, A.H. Stroud published his encyclopedic work on multidimensional numerical integration, see [20]. In this book Stroud presented a comprehensive bibliography and listing of almost all cubature formulae, known at the time, for a variety of regions of integration. Continuing the work of Stroud, Cools and Rabinowitz see [57] [58] have published a reference to the collection of all known cubature formulae, which have appeared in publication since Stroud.

A vast variety of techniques has been explored in order to produce cubature formulae with a small number of nodes, among the most notable efforts are the measure theoretic, randomised and sparse grid methods. Prior to illustrating these, we will recall some other significant approaches.

The invariant theoretic approach exploits the symmetry of many common multidimensional domains of integration such as the unit hypercube, unit sphere, unit simplex, etc. Sobolev, in [35] has shown how to simplify the moment matching equations in the definition (1.0.11), when considering the non-linear formulation of the problem by using the theory of invariant polynomials. This approach has been extended by Espelid [31], [32], Genz [34], Haegemans and Cools [33]. This approach, however does not remove the need of solving nonlinear systems of equations, that for moderately large dimensions and degrees are still too complex to be solved in closed form. Another notable method of constructing cubature formulae with minimal number of nodes is based on ideal theory and has been employed in [37], [38],[39]. Whilst useful for practical cubature construction in 2 dimensions, higher dimensional results have not yet been obtained. Another approach is that of lattice rules, [14] which are particularly suitable for numerical integration of periodic functions. However, if one wants to obtain a high rate of convergence for smooth non-periodic functions, then one has to transform the integral into a periodic function. These transformations often reduce the smoothness of the underlying functions and drastically increase the norms of their derivatives.

In the following, we will introduce randomised or Monte Carlo numerical integration methods and appealing to their underlying limitations we then proceed onto Quasi Monte Carlo methods, based on equidistributed and low-discrepancy sequences of nodes, this constitutes the number-theoretic approach to cubature. Next, we shall illustrate the Sparse Grid methods, based on the Smolyak algorithm. We will present extensive results for this methodology as it will, in the coming chapters, serve as a comparison for the cubature algorithm developed in this thesis: Carathéodory Cubature

Algorithm. Finally we will outline the key ideas behind the Carathéodory Cubature Algorithm.

1.2.2 Monte Carlo Methods

In very high dimensions, $d \geq 15$ all numerical cubature constructions become too expensive, so randomised methods, also known as Monte-Carlo methods, which do not suffer from the "curse of dimensionality", are often employed as their convergence rate is independent of dimension. However, we note that the computational effort ofMonte-Carlo methods, is still highly dependent on the underlying dimension of the domain of integration.

Monte Carlo numerical integration is based on probabilistic interpretation of the integral (1.15). Consider a sequence $\{\underline{\boldsymbol{x}}_n\}_{n=1}^N \subseteq [0,1]^d$ of uniformly distributed pseudo-random numbers and equal weights $\omega_i = 1/N$. Then, by the Strong Law of Large of Numbers (Feller 1971), the empirical expectation

$$MC_N[f] := \frac{1}{N}\sum_{n=1}^{N} f(\underline{\boldsymbol{x}}_n)$$

converges to $I_d[f]$ with probability one, as $n \to \infty$ if f has finite variance $\sigma^2(f)$, where

$$\sigma(f) = \left(\int_{[0,1]^d} \big(f(\underline{\boldsymbol{x}}) - I_d[f]\big)^2 d\underline{\boldsymbol{x}}\right)^{1/2}$$

The Central Limit Theorem tells us that the Monte Carlo integration error is distributed as $\epsilon^{MC} \sim \sigma(f)N^{-\frac{1}{2}}X$, where $X \sim N(0,1)$ is a standard normal random variable. Hence, the MC error is of size $O(N^{-\frac{1}{2}})$, with a constant given by the variance of f. In contrast to deterministic integration methods, we do not have an upper bound on the absolute integration error, but rather we obtain a result that tells us that the error is of a certain size with some probability.

The computational time for Monte-Carlo is proportional to the number of samples, which grows rapidly as the desired accuracy is tightened, indeed

$$\epsilon^{MC} = O\big(\sigma(f)N^{-\frac{1}{2}}\big)$$
$$N = O\left(\frac{\sigma(f)}{\epsilon^{MC}}\right)$$

There are many options for accelerating Monte-Carlo or equivalently reducing it's error. In the following, we present two such methods, for more details see [18]. The first is based on variance reduction, in which the integrand is transformed so as to reduce the constant variance $\sigma(f)$. The second is to replace the actual sampling random variables by an alternative sequence which improves the exponent of convergence $1/2$. The first approach includes methods such as stratified sampling, control variates and importance sampling, which we briefly review in the following.

1.2.2.1 Stratified Sampling

Stratification consists of splitting the domain of integration R into sub-domains R_j, performing Monte-Carlo on each sub-domain and adding the results, to obtain the following

$$MC_N^S[f] := \sum_{j=1}^{M} \frac{|R_j|}{n_j} \sum_{i=1}^{n_j} f(\underline{\boldsymbol{x}}_i^{(j)})$$

where $\underline{\boldsymbol{x}}_i^{(j)}$ are points in R_j. The variance and error of the stratified Monte-Carlo become

$$\sigma_S^2(f) = \sum_{j=1}^{M} \frac{|R_j|}{n_j} \sigma_j^2(f), \qquad \epsilon_S = O\Big(\sigma_S(f) N^{-\frac{1}{2}}\Big)$$

where the variance over each sample R_j, $\sigma_j^2(f)$ is given by

$$\sigma_j^2(f) = \frac{1}{|R_j|} \int_{R_j} \left(f(\underline{\boldsymbol{x}}) - \frac{1}{|R_j|} \int_{R_j} f(\underline{\boldsymbol{x}}) \right)^2 d\underline{\boldsymbol{x}}$$

Since the variance over a subset is always less than the variance over the whole set, we have $\sigma_S \leq \sigma$. Thus, stratification always lowers the integration error.

1.2.2.2 Importance Sampling

Importance sampling technique consists of introducing a density p

$$I[f] = \int f(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}} = \int \frac{f(\underline{\boldsymbol{x}})}{p(\underline{\boldsymbol{x}})} p(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}}$$

Now, we consider the integral $I[f]$ as an integral with respect to density p and sample points from this density, to form the Monte-Carlo estimate

$$MC_N^p[f] := \frac{1}{N} \sum_{n=1}^{N} \frac{f(\underline{\boldsymbol{x}}_n)}{p(\underline{\boldsymbol{x}}_n)}$$

the resulting variance and error are given by

$$\sigma_p^2(f) = \int \left(\frac{f(\underline{\boldsymbol{x}})}{p(\underline{\boldsymbol{x}})} - I[f] \right)^2 p(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}}, \qquad \epsilon_p = O\Big(\sigma_p(f) N^{-\frac{1}{2}}\Big)$$

Hence, importance sampling is effective if we choose f/p is nearly constant, so that σ_p is small. Since if we choose $p(\underline{\boldsymbol{x}}) = cf(\underline{\boldsymbol{x}})$ with $c = \frac{1}{\int f(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}}}$, then $p(\underline{\boldsymbol{x}})$ normalises to unity

$$\int p(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}} = 1$$

giving an estimation to the integral with variance zero. In practice it is often difficult to find an appropriate density p such that f/p is nearly constant, and sample variables from this density. A more successful application of this method is to emphasise some rare but important events, that is small regions of the domain of integration where f is large.

1.2.2.3 Control Variates

The idea of control variates is to use an integrand g which approximates f and for which the integral $I[g] = \int g(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}}$ is known. By linearity of the integral, we have

$$\int f(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}} = \int \Big(f(\underline{\boldsymbol{x}}) - g(\underline{\boldsymbol{x}}) \Big) + \int g(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}}$$

If the function g is chosen carefully, the variance of $(f - g)$ will be smaller than the variance of f and the Monte-Carlo estimate constructed on the integral of $(f - g)$, will have a reduced variance.

Finally, we note that the convergence rate of Monte Carlo does not take into account the smoothness of the underlying function. Thus, smoother integrals are not computed more efficiently than the non-smooth ones. As noted in [18], comparing the rates of convergence of Monte Carlo with product cubatures for a given function of smoothness r, $f \in C^r[0,1]^d$, Monte Carlo method performs better if $r/d < 1/2$. That is, Monte Carlo converges faster for low smoothness, high dimensional integrals. By Bahvalov's result (1.19), if $r = 1$, then for any dimension $d > 2$, the best one can say about the error of any non-probabilistic cubature formula is that it would be $O(N^{-\frac{1}{d}})$, which yields a convergence rate slower than Monte Carlo. In one dimension, more rapid convergence than Monte-Carlo is easily obtained, but in multiple dimensions and for discontinuous integrands or integrands containing singularities Monte-Carlo is a very efficient integration method.

1.2.3 Quasi Monte Carlo

One of the drawbacks of the Monte-Carlo methods is that some areas of the domain of integration are sampled heavily, whilst other areas are not sampled at all, a phenomenon known as clustering. In contrast to pseudo-random sequences of random variables, employed by Monte-Carlo, quasi-random sequences are designed to provide better uniformity throughout the domain of integration and hence faster convergence. Quasi-random, also known as low-discrepancy sequences of abscissas, form the basis of a number-theoretic approach to multidimensional integration.

In the following we will describe the uniformity of a sequence of points in terms of its discrepancy, to introduce this we need to define the following

For any sequence of points $\{\underline{\boldsymbol{x}}_n\}_{n=1}^N$ in $[0,1]^d$, define

$$R_N(E) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{\{\underline{\boldsymbol{x}}_n \in E\}} - vol(E)$$

for any subset E of $[0,1]^d$, where $\mathbb{1}_{\{A\}}$ denotes the indicator function. The discrepancy of a sequence is thereby defined by some norm applied to $R_N(E)$ for an appropriately chosen set of subsets of $[0,1]^d$, such as the collection of measurable subsets of $[0,1]^d$.

Definition 1.2.7. *[Definition 6.3.2 [2]] For a non-empty set $\mathcal{M}$ of measurable subsets of $[0,1]^d$, the discrepancy $\mathcal{D}_N^{\mathcal{M}}$ of a sequence $\{\underline{\boldsymbol{x}}_n\}_{n=1}^N$ in $[0,1]^d$ with respect to $\mathcal{M}$ is defined by*

$$\mathcal{D}_N^{\mathcal{M}}(\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_N) := \sup_{E \in \mathcal{M}} \left| R_N(E) \right|$$

The two following choices of $\mathcal{M}$

$$\mathcal{M}_1 = \left\{ [a_1, b_1) \times \ldots \times [a_d, b_d) \subseteq [0,1]^d \right\}$$

$$\mathcal{M}_2 = \left\{ [0, b_1) \times \ldots \times [0, b_d) \subseteq [0,1]^d \right\}$$

lead to discrepancies $\mathcal{D}_N$ and $\mathcal{D}_N^*$ respectively, the latter is known as the star discrepancy.

The following theorem illustrates what it means for a sequence to be equidistributed or uniformly distributed.

Theorem 1.2.8. *[Theorem 6.3.2 [2]] An infinite sequence* $\{\underline{\boldsymbol{x}}_n\}_{n=1}^{\infty}$ *in* $[0,1]^d$ *is equidistributed if and only if*

$$\lim_{n\to\infty} \mathcal{D}_N(\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_N) = 0$$

Moreover, the sequence $\{\underline{\boldsymbol{x}}_n\}_{n=1}^{\infty}$ is called as quasi-random, see [18] if

$$D_N(\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_N) \leq c(log(N))^k N^{-1}$$

where c, k are constants independent of N, but may depend on dimension d. The simplest example of a quasi-random sequence for $d = 1$ is Van der Corput sequence, obtained by reverting the binary bits of a number n around the decimal points as follows

$$n = a_m a_{m-1} \ldots a_1 a_0$$

$$x_n = 0.a_0 a_1 \ldots a_{m-1} a_m$$

Notable d-dimensional low-discrepancy sequences, for which the discrepancy is small, are the quasi-random sequences Halton[23], Sobol [24], Niederreiter [25]. All of these sequences exhibit the following discrepancy bound

$$D_N = O((logN)^d N^{-1})$$

The basis for the error analysis of Monte-Carlo integration based on quasi-random sequences is the Koksma-Hlawka inequality.

Theorem 1.2.9. *[Theorem 5.1 [18]] For any sequence* $\{\underline{\boldsymbol{x}}_n\}_{n=1}^{N}$ *in* $[0,1]^d$ *and any function* f *of bounded variation, let the Quasi Monte-Carlo estimate and it's corresponding error be given by*

$$MC_N^Q[f] := \frac{1}{N}\sum_{n=1}^{N} f(\underline{\boldsymbol{x}}_n), \qquad \epsilon[f] = \Big| MC_N^Q[f] - I[f] \Big|$$

then the following error bound holds

$$\epsilon[f] \leq V[f]\mathcal{D}_N^*$$

where $V[f]$ *is the variation of the function* f *in the Hardy-Krause sense, that is*

$$V[f] = \int_{[0,1]^d} \Big| \frac{\partial^d f}{\partial t_1 \ldots \partial t_d} \Big| dt_1 \ldots dt_d + \sum_{i=1}^{d} V[f_1^{(i)}]$$

in which $f_1^{(i)}$ *is the restriction of the function* f *to the boundary* $x_i = 1$

Hence for any quasi-random sequence with discrepancy of order $O((logN)^d N^{-1})$, the Koksma-Hlawka inequality implies that the Quasi Monte-Carlo integration error is bounded by

$$\epsilon[f] \leq cV[f]((logN)^d N^{-1}$$

And hence the Quasi Monte-Carlo method for the aforementioned low-discrepancy quasi-random sequences achieves the reduction in the exponent of N from $1/2$ to 1, when compared to standard Monte-Carlo. However, we note that a dependence on dimension is introduced through the $(logN)^d$ term. For large dimension d, the bound is dominated by this term, unless $N > 2^d$. Moreover, Quasi Monte-Carlo integration is found to lose its effectiveness if the integrand f is not smooth. The factor $V[f]$ in the upper bound is an indicator of this dependence, although in practice a much smaller amount of smoothness, somewhere between continuity and differentiability, is usually sufficient, see more details in [18].

1.2.4 Smolyak Cubature

In 1963, Smolyak [40] derived a construction of multivariate cubature formulae by providing a suitable linear combination of tensor products of univariate quadrature formulae, utilising a considerable smaller number of function evaluations than the full cubature tensor product (1.16). Hence, this method is also known as the Sparse Grid method. Smolyak considered the following class of functions, with bounded mixed derivatives up to order r

$$\mathcal{W}_d^r = \left\{ f:\ [0,1]^d \to \mathbb{R}, \quad \left\| \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}} \right\|_\infty < \infty,\ \alpha = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}_0^d,\ \alpha_i \leq r \right\}$$

Assume that $\{Q_i\}_{i\geq 1}$ is a collection of n_i-node quadrature formulae on $[0,1]$ where Q_i is given by

$$Q_i[f] = \sum_{j=1}^{n_i} \omega_j^i f(x_j^i) \qquad \omega_j^i \in \mathbb{R},\ \left(x_j^i\right)_{j=1}^n \subset [0,1] \quad \text{for} \quad i \geq 1$$

Now, utilising the notation $\Delta_i = (Q_i - Q_{i-1})$, for $i \in \mathbb{N}$, with $Q_0 = 0$. The ℓ^{th} level Smolyak cubature formula is given by

$$Q^{SM}(\ell, d) = \sum_{|i| \leq \ell + d - 1} (\Delta_{i_1} \otimes \dots \otimes \Delta_{i_d})$$

with $i = (i_1, \dots, i_d) \in \mathbb{N}_0^d$ and $|i| = |i|_1$. Hence, the summation is taken over the simplex $|i|_1 \leq \ell + d - 1$, in contrast to a full tensor cubature product formula

$$\left(Q_\ell \otimes \dots \otimes Q_\ell\right) = \sum_{j=1}^{d} \sum_{1 \leq i_j \leq \ell} (\Delta_{i_1} \otimes \dots \otimes \Delta_{i_d})$$

where the summation is taken over the cube $|i|_\infty \leq \ell$. For example, for case $d = 2$, we have

$$Q^{SM}(\ell, 2) = \sum_{i_1 + i_2 = \ell} (Q_{i_1} \otimes Q_{i_2}) - \sum_{i_1 + i_2 = \ell - 1} (Q_{i_1} \otimes Q_{i_2})$$

and for a general dimension d the analogous formula is

$$Q^{SM}(\ell, d) = \sum_{\ell \leq |i| \leq \ell + d - 1} (-1)^{\ell + d - |i| - 1} \cdot \binom{d-1}{|i| - 1} \cdot (Q_{i_1} \otimes Q_{i_2} \otimes \dots \otimes Q_{i_d})$$

Smolyak construction comes in various flavours depending on the constituent sequence

of one-dimensional quadratures. Smolyak Algorithm with Clenshaw-Curtis quadratures is studied in [62], with Gauss-Legendre and Gauss-Patterson in [93] and with Gauss-Hermite [45].

The error bound for Smolyak cubature based on the sequence of interpolatory quadratures with positive weights, such as Clenshaw-Curtis, Gauss-Legendre and Gauss-Patterson, is given by

$$\left|R_d^n[f]\right| = \left|I_d[f] - Q^{SM}(\ell, d)[f]\right| = O\Big(n^{-r} \cdot (logn)^{(d-1)\cdot(r+1)}\Big)$$

for $f \in \mathcal{W}_d^r$. If f has mixed bounded derivatives up to and including order r, the above estimate is optimal, up to a logarithmic factor, in the smoothness scale considered, for any $r \in \mathbb{N}$, see [92]. For classical spaces $\mathcal{C}^r[0,1]^d$ error bounds for Smolyak construction can also be derived, see [44], but they indicate exponential dependence on dimension.

Clearly, $Q^{SM}(\ell, d)$ is a linear functional and it depends on f only through function evaluations on a finite number of points. To describe these points, let

$$supp(Q_i) = \{x_1^i, \ldots, x_{n_i}^i\} \subset [0,1]$$

denote the set of nodes of Q_i. The full tensor product $(Q_{i_1} \otimes Q_{i_2} \otimes \ldots \otimes Q_{i_d})$ is based on the grid $supp(Q_{i_1}) \times \ldots \times supp(Q_{i_d})$ and therefore $Q^{SM}(\ell, d)[f]$ depends at most on the values of f at the union

$$H(\ell, d) = \bigcup_{|i| \leq \ell + d - 1} \Big(supp(Q_{i_1}) \times \ldots \times supp(Q_{i_d})\Big) \subset [0,1]^d \tag{1.27}$$

Clearly, quadrature sequences with nested support, $supp(Q_i) \subset supp(Q_{i+1})$, are the most economical choice for the Smolyak construction, as they yield $H(\ell, d) \subset H(\ell + 1, d)$ and hence the number of functions evaluations in (1.27) is reduced to the sparse grid

$$\bigcup_{|i| = \ell + d - 1} \Big(supp(Q_{i_1}) \times \ldots \times supp(Q_{i_d})\Big) \subset [0,1]^d$$

The number of points $n(\ell, d)$ in the sparse grid $H(\ell, d)$ clearly satisfies

$$n(\ell, d) \leq n(H(\ell, d)) = \sum_{|i| \leq \ell + d - 1} n_{i_1} \ldots n_{i_d}$$

and if we further assume that $n_i = O(2^i)$, then the order of $n(\ell, d)$, then as noted in [92][lemma 5], we have

$$n(\ell, d) = O(2^\ell \cdot \ell^{d-1})$$

in contrast with full tensor product rules, which would give $O(2^{\ell d})$ number of points.

The ℓ^{th} level Smolyak Algorithm when applied to f can be written as

$$Q^{SM}(\ell, d)[f] = \sum_{|i| \leq \ell+d-1} \sum_{j_1=1}^{n_{i_1}} \dots \sum_{j_d=1}^{n_{i_d}} \lambda_j^i \cdot f(x_{j_1}^{i_1}, \dots, x_{j_d}^{i_d}) \tag{1.28}$$

where, in the case the quadratures are nested, the weights are given by

$$\lambda_j^i = \sum_{|i+q| \leq \ell+2d-1} \beta_{(i_1+q_1)j_1} \dots \beta_{(i_d+q_d)j_d}$$

with $q \in \mathbb{N}^d$, where

$$\beta_{(i+q)j} = \begin{cases} \omega_j^i & \text{if} \quad q = 1 \\ \omega_r^{k+q} - \omega_s^{k+q-1} & \text{if} \quad q > 1, \ x_j^i = x_r^{k+q-1} = x_s^{k+q-2} \end{cases}$$

and in the non-nested case the weights are given by $\lambda_j^i = \omega_{j_1}^{i_1} \dots \omega_{j_d}^{i_d}$, see [93].

Regardless of the positivity or non-positivity of the underlying sequence of quadrature rules in the Smolyak construction, the resulting algorithm will contain negative weights, hence it is especially important to avoid numerical cancellation, so instead of implementing the summation in (1.28), with increasing ℓ, it is recommended to perform summation coordinate-wise, see [93]. Convergence is in general guaranteed, due to the absolute values of the weights remaining relatively small. However, it can be shown [44], that both in the nested and in the non-nested case the norm of $Q^{SM}(\ell, d)$

$$\|Q^{SM}(\ell, d)\| = \sum_{|i| \leq \ell+d-1} \sum_{j_1=1}^{n_{i_1}} \dots \sum_{j_d=1}^{n_{i_d}} |\lambda_j^i| = O\big((log(n(\ell, d)))^{d-1}\big) \tag{1.29}$$

Hence $\|Q^{SM}(\ell, d)\|$ exhibits exponential growth in the log of its cardinality. Thus the norm may be very large, rendering the Smolyak algorithm unstable. The norms of $Q^{SM}(\ell, d)$ are generally much smaller if the underlying sequence of quadratures are Clenshaw-Curtis formulae, rather than Gaussian quadrature formulae.

Letting Π_ℓ denote the space of all one-dimensional polynomials of degree $\leq \ell$, we consider the following polynomial spaces

$$\mathcal{P}_\ell^d := \Big\{ \Pi_{i_1} \otimes \dots \otimes \Pi_{i_d}, \quad |i| = \ell + d - 1 \Big\}$$

If Q_ℓ is exact for Π_ℓ, then the ℓ^{th} level Smolyak cubature $Q^{SM}(\ell, d)$ is exact for $\mathcal{P}_\ell^d$, see [93].

However, in order to obtain an estimate in terms of the classical polynomial exactness for a cubature formula, based on the space of all d-variate polynomials of degree $\leq \ell$, that is

$$\Pi_\ell^d := \left\{ \Pi_{i_1} \otimes \cdots \otimes \Pi_{i_d}, \quad |i|_\infty = \ell \right\}$$

we have the following lemma.

Lemma 1.2.10. *([42]) Let $\{Q_i\}_{i\in\mathbb{N}}$ be a sequence of quadrature formulae, if for all $i \in \mathbb{N}$*

$$deg(Q_i) \geq 2i - 1$$

Then,

$$deg(Q^{SM}(\ell, d)) \geq 2\ell + 1$$

Many sequences of quadrature formulae can be constructed to satisfy the assumption of the aforementioned lemma, amongst them: Clenshaw-Curtis, Gauss-Patterson, Gauss-Legendre.

1.2.4.1 Clenshaw-Curtis Sparse Grids

Let $\{Q_i^{CC}\}_{i\geq 1}$ denote a collection of n_i-node Clenshaw-Curtis quadrature formulae on $[0, 1]$, where Q_i is given by

$$Q_i^{CC}[f] = \sum_{j=1}^{n_i} \omega_j^i f(x_j^i) \qquad \omega_j^i \in \mathbb{R}, \ \left(x_j^i\right)_{j=1}^n \subset [0,1] \quad \text{for} \ \ i \geq 1$$

and the abscissas are given by the extrema of Chebyshev polynomials, as described in Section 1.1.2, that is

$$x_j^i = -cos\left(\frac{\pi \cdot (j-1)}{n_i - 1}\right) \qquad j = 1, \ldots, n_i$$

similarly the corresponding weights are provided in Section 1.1.2. As noted in [62], sparse grids based on a sequence of Clenshaw-Curtis quadrature formulae are often

preferable to other sparse grids, as they are nested and the weights of different signs at common points partially cancel each other, which ensures superior stability of the cubatures generated.

In order to obtain nested sets of abscissa from a sequence of Clenshaw-Curtis quadrature formulae, we need to choose Q_i^{CC} such that, see [46]

$$n_i = 2^{i-1} + 1 \qquad \text{for} \quad i > 1 \qquad \text{and} \quad n_1 = 1 \tag{1.30}$$

In light of (1.6), Clenshaw-Curtis quadrature formulae satisfy the following error bound

$$\left|R^n[f]\right| = \left|I[f] - Q_i^{CC}[f]\right| \leq \gamma_r \cdot 2^{-r \cdot i} \tag{1.31}$$

where γ_r is a constant dependant on r. Then the error of the ℓ^{th} level Smolyak cubature for integrands $f \in \mathcal{W}_d^r$ is illustrated in the following theorem.

Theorem 1.2.11. *[Theorem 1 [62]] Let* $\theta_r = \max\left(2^{r+1}, \gamma \cdot (1 + 2^r)\right)$*, then the error of the Smolyak cubature* $Q_{CC}^{SM}(\ell, d)$ *satisfies*

$$\left|R_d^n[f]\right| = \left|I_d[f] - Q_{CC}^{SM}(\ell, d)[f]\right| \leq \gamma_r \cdot {\theta_r}^{d-1} \cdot \binom{\ell}{d-1} \cdot 2^{-r\ell}$$

We note that since $deg(Q_i^{CC}) = n_i - 1 = 2^{i-1} \geq 2i - 1$ the requirement of lemma (1.2.10) is satisfied, therefore we have $deg(Q_{CC}^{SM}(\ell, d)) \geq 2\ell + 1$. As noted in [47], the precision of the Smolyak sparse grids grows linearly with the level ℓ, whereas the precision and the number of nodes in the constituent sequence of Clenshaw-Curtis formulae grows exponentially with dimension. This an anomaly, that suggests that there may be a more economical way of constructing sparse grids from Clenshaw-curtis formulae.

1.2.4.2 Delayed Clenshaw-Curtis Sparse Grids

The usual sparse grid construction of level ℓ is only guaranteed to achieve a precision of $2\ell+1$ for Clenshaw-Curtis sparse grid, but if we construct the sparse grid based on the sequence of Clenshaw-Curtis formulae satisfying (1.30), we will greatly exceed the precision requirement by obtaining an accuracy of 2^ℓ for some higher order monomials. We

do so at a corresponding higher cost, as noted in [47]. To avoid this overhead, we note that the Smolyak construction does not require successive members of the sequences of quadratures to be distinct, and it is possible to base the sparse grid on a modified sequence $\tilde{Q}_i$ with the number of nodes $\tilde{n}_i$ increasing slower with i, than n_i, which will come close to achieving linear growth in the cardinality of the quadrature sequence $\tilde{Q}_i$.

The sequence n_i of number of nodes in Clenshaw-Curtis quadrature formulae Q_i takes the following form

$$\{n_i\}_{i\geq 1} = \{1, 3, 5, 9, 17, 33, \ldots, 2^{i-1} + 1, \ldots\}$$

If we require in the Smolyak construction a sequence of quadratures such that at each specific level ℓ the minimum polynomial degree $2\ell+1$ is reached, then we need to select quadrature formulae that are not necessarily distinct. For example, for levels 0 to 4 the number of nodes in the above family is $1, 3, 5, 9, 17$ and the corresponding minimum precision requirements are $1, 3, 5, 7, 9$, hence $n_3 = 9$ exceeds the precision level required, moreover it is still good enough to meet the precision at level 4. Following this logic, we obtain the following sequence of cardinalities of $\{\tilde{Q}_i\}_{i\geq 1}$

$$\{\tilde{n}_i\}_{i\geq 1} = \{1, 3, 5, 9, 9, 17, 17, 17, 17, 33 \ldots\}$$

In general, the sparse grids based on Clenshaw-Curtis family $\{Q_i\}_{i\geq 1}$ and the delayed Clenshaw-Curtis family $\{\tilde{Q}_i\}_{i\geq 1}$ will be the same until level 4, but at level 4 and beyond the difference in the number of points is very noticeable, particularly for low-dimensional grids. For illustration below we present the number of nodes n_i and $\tilde{n}_i$ in dimensions 2 and 3 up to level 10, for more details see [47]

DIM	2	3	DIM	2	3
SMOLYAK CC LEVEL			SMOLYAK CC DEL LEVEL		
0	1	1	0	1	1
1	5	7	1	5	7
2	13	25	2	13	25
3	29	69	3	29	69
4	65	177	4	49	153
5	145	441	5	81	297
6	321	1073	6	129	545
7	705	2561	7	161	881
8	1537	6017	8	225	1361
9	3329	13953	9	257	1953
10	7169	32001	10	385	2721

Table 1.1: Smolyak Clenshaw-Curtis and Delayed Smolyak Clenshaw-Curtis

1.3 Contributions and Significance

1.3.1 Carathéodory Cubature Problem Formulation

Given

$$I_d[f] = \int_{[0,1]^d} f(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}}, \qquad \underline{\boldsymbol{x}} = (x^1, \dots, x^d) \tag{1.32}$$

we will provide a constructive derivation of an interpolatory cubature formula $Q^{CAR}(m,d)$, whose existence is proved in Tchakaloff's Theorem, satisfying the desired properties of cubature formula (1.24).

From Section 1.2, we recall that by definition, any cubature formula of degree $m \in \mathbb{N}$, is required to integrate all d-variate polynomials of degree $\leq m$ exactly. To achieve this, we may proceed by fixing a collection of abscissae $\{\underline{\boldsymbol{x}}_1, \dots, \underline{\boldsymbol{x}}_n\}$ and form a linear system of moment matching equations. In the following, we choose the nodes $\{\underline{\boldsymbol{x}}_1, \dots, \underline{\boldsymbol{x}}_n\}$, to be the nodes arising form the Cartesian product of d-univariate quadrature formulae. Let $\{Q_i\}_{i=1}^d$ denote a collection of d univariate quadratures of degree $m \in \mathbb{N}$

$$Q[f] := Q_i[f] = \sum_{j=1}^{n} \lambda_j^i f(x_j^i), \tag{1.33}$$

with

$$\lambda_j^i \in \mathbb{R}_+, \ \sum_{i=1}^{n} \lambda_j^i = 1, \ \left\{x_j^i\right\}_{j=1}^n \subset [0,1] \quad \text{for} \ \ i \in [\![1,d]\!] \tag{1.34}$$

for example we could take the Clenshaw-Curtis quadrature formulae with $n = m+1$ or the Gauss-Legendre quadrature formulae with $n = \left(\lfloor \frac{m}{2} \rfloor + 1\right)$, yielding the Cartesian product cubature of degree $m \in \mathbb{N}$

$$Q(m,d)[f] := \sum_{j_1=1}^{n} \dots \sum_{j_d=1}^{n} \lambda_{j_1}^1 \dots \lambda_{j_d}^d f(x_{j_1}^1, \dots, x_{j_d}^d) = \sum_{i=1}^{N} \omega_i f(\underline{\boldsymbol{x}}_i) \tag{1.35}$$

$$\text{where} \quad \omega_i = \lambda_{j_1}^1 \dots \lambda_{j_d}^d \in \mathbb{R}_+, \ \sum_{i=1}^{N} \omega_i = 1, \ \left\{\underline{\boldsymbol{x}}_i\right\}_{i=1}^N \subset [0,1]^d, \ N = n^d$$

By definition of a cubature formula, $Q(m,d)$ integrates exactly all the basis elements $\mathcal{B}(\Pi_m^d) = \{\gamma_1, \dots, \gamma_M\}$ for some fixed basis for Π_m^d with $\gamma_1(\underline{\boldsymbol{x}}) = 1$ and $dim(\Pi_m^d) = \binom{m+d}{d} := M$. Hence, the nodes and weights of $Q(m,d)$ satisfy the following system of moment matching equations

$$\omega_1\gamma_1(\underline{\boldsymbol{x}}_1) + \ldots + \omega_N\gamma_1(\underline{\boldsymbol{x}}_N) = \int_{[0,1]^d} \gamma_1(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}}$$
$$\omega_1\gamma_2(\underline{\boldsymbol{x}}_1) + \ldots + \omega_N\gamma_2(\underline{\boldsymbol{x}}_N) = \int_{[0,1]^d} \gamma_2(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}}$$
$$\vdots$$
$$\omega_1\gamma_M(\underline{\boldsymbol{x}}_1) + \ldots + \omega_N\gamma_M(\underline{\boldsymbol{x}}_N) = \int_{[0,1]^d} \gamma_M(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}}$$

By interpreting the moment matching equations above, as a linear system in variables $\{\omega_1, \ldots, \omega_n\}$, we can write concisely

$$A\underline{\boldsymbol{\omega}} = \underline{\boldsymbol{b}}, \quad \underline{\boldsymbol{\omega}} > \underline{\mathbf{0}} \tag{1.36}$$

where $A \in \mathbb{R}^{M\times N}, \quad \underline{\boldsymbol{\omega}} \in \mathbb{R}^N_+, \quad \underline{\boldsymbol{b}} \in \mathbb{R}^M$ are given by

$$A = \begin{bmatrix} \gamma_1(\underline{\boldsymbol{x}}_1) & \ldots & \gamma_1(\underline{\boldsymbol{x}}_N) \\ \vdots & \vdots & \vdots \\ \gamma_M(\underline{\boldsymbol{x}}_1) & \ldots & \gamma_M(\underline{\boldsymbol{x}}_N) \end{bmatrix}, \quad \underline{\boldsymbol{\omega}} = \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_N \end{bmatrix}, \quad \underline{\boldsymbol{b}} = \begin{bmatrix} \int_{[0,1]^d} \gamma_1(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}} \\ \vdots \\ \int_{[0,1]^d} \gamma_M(\underline{\boldsymbol{x}})d\underline{\boldsymbol{x}} \end{bmatrix}$$

and $\sum_{i=1}^N \omega_i = 1$ is given, since $\gamma_1(\underline{\boldsymbol{x}}) = 1$. Now, by defining the (m, d)-monomial mapping $\gamma_m^d := (\gamma_1, \ldots, \gamma_M) : [0, 1]^d \to \mathbb{R}^M$, we can write

$$A = [\gamma_m^d(\underline{\boldsymbol{x}}_1), \cdots, \gamma_m^d(\underline{\boldsymbol{x}}_N)]$$

and by (1.36), we have

$$\underline{\boldsymbol{b}} = A\underline{\boldsymbol{\omega}} = \sum_{i=1}^N \omega_i\gamma_m^d(\underline{\boldsymbol{x}}_i) \in Conv\Big(\{\gamma_m^d(\underline{\boldsymbol{x}}_1), \cdots, \gamma_m^d(\underline{\boldsymbol{x}}_N)\}\Big)$$
$$:= \left\{ \sum_{i=1}^N \alpha_i\gamma_m^d(\underline{\boldsymbol{x}}_i) \;\middle|\; \forall_i : \alpha_i \geq 0 \;\wedge\; \sum_{i=1}^N \alpha_i = 1 \right\}$$

where $Conv(\{\underline{\boldsymbol{z}}_1, \ldots, \underline{\boldsymbol{z}}_N\})$ denotes the Convex Hull of the set $\{\underline{\boldsymbol{z}}_1, \ldots, \underline{\boldsymbol{z}}_N\}$.

We recall the Carathéodory Convex Hull Theorem

Theorem 1.3.1. *Carathéodory Convex Hull Theorem[p263-264 [51]]*
Let $S \subset \mathbb{R}^M$, $card(S) > M+1$, if $\underline{\boldsymbol{x}} \in Conv(S)$ the convex hull of S, then $\underline{\boldsymbol{x}} \in Conv(R)$, for some $R \subset S$, $card(R) \leq M+1$.

The construction of the Cartesian product cubature of degree $m \in \mathbb{N}$ in (1.35), yields a collection of nodes $\{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_N\}$ and strictly positive weights $\{\omega_1, \ldots, \omega_N\}$. By considering the span of the columns of the matrix A in (1.36), given by the (m,d)-monomial map evaluations at the nodes $\{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_N\}$, we set

$$S := \{\gamma_m^d(\underline{\boldsymbol{x}}_1), \cdots, \gamma_m^d(\underline{\boldsymbol{x}}_N)\} \subseteq \mathbb{R}^M$$

For large enough degree $m \in \mathbb{N}$ and dimension $d \in \mathbb{N}$, we will have $card(S) = N > M+1$. Thus, by appealing to the Carathéodory Convex Hull Theorem there exists a subsequence

$$R := \{\gamma_m^d(\underline{\boldsymbol{x}}_{i_1}), \cdots, \gamma_m^d(\underline{\boldsymbol{x}}_{i_{M'}})\} \subset S, \qquad M' \leq M+1$$

satisfying

$$\underline{\boldsymbol{b}} = \sum_{i=1}^{N} \omega_i \gamma_m^d(\underline{\boldsymbol{x}}_i) \in Conv\Big(\{\gamma_m^d(\underline{\boldsymbol{x}}_{i_1}), \cdots, \gamma_m^d(\underline{\boldsymbol{x}}_{i_{M'}})\}\Big) \subseteq \mathbb{R}^M$$

The above is equivalent to stating that there exists a collection of weights $\{\widehat{\omega}_1, \ldots, \widehat{\omega}_{M'}\} \subset \mathbb{R}_+$, for which we have

$$\underline{\boldsymbol{b}} = \sum_{k=1}^{M'} \widehat{\omega}_k \gamma_m^d(\underline{\boldsymbol{x}}_{i_k}) \tag{1.37}$$

We note that $\sum_{k=1}^{M'} \widehat{\omega}_k = 1$, is implicitly given by (1.37), since $\gamma_1(\underline{\boldsymbol{x}}) = 1$.

Once we know how to obtain the collection $\{\widehat{\omega}_k\}_{k=1}^{M'}$, we proceed to select M' nodes $\{\underline{\boldsymbol{x}}_{i_k}\}_{k=1}^{M'}$ out of N product cubature nodes $\{\underline{\boldsymbol{x}}_i\}_{i=1}^{N}$ in (1.35), receiving strictly positive weights $\widehat{\omega}_k$.

We recall that the original problem of finding an interpolatory cubature, consists of finding a collection of nodes $\{\underline{\boldsymbol{x}}_{i_k}\}_{k=1}^{M'}$ of cardinality $M' \leq \binom{m+d}{d}$ and a sequence of weights $\{\widehat{\omega}_k\}_{k=1}^{M'}$ which solves the moment matching equations for the prescribed collection of nodes uniquely. This problem can now be re-formulated as follows: given a collection of cubature weights $\{\omega_i\}_{i=1}^{N}$ and nodes $\{\underline{\boldsymbol{x}}_i\}_{i=1}^{N}$ of some cardinality $N > M := \binom{m+d}{d}$ we endeavour to find a new sequence of positive weights $\{\widehat{\omega}_k\}_{k=1}^{M'}$

on the subset $\{\underline{\boldsymbol{x}}_{i_k}\}_{k=1}^{M'}$ and formally zero weights on $\{\underline{\boldsymbol{x}}_i\}_{i=1}^{N} \backslash \{\underline{\boldsymbol{x}}_{i_k}\}_{k=1}^{M'}$. Thus, by finding such a collection of strictly positive weights $\{\widehat{\omega}_k\}_{k=1}^{M'}$, we simultaneously find the weights and the nodes of an interpolatory cubature formula.

Now we are in a position to form an interpolatory cubature formula of degree $m \in \mathbb{N}$, which we will denote by

$$Q^{CAR}(m,d)[f] = \sum_{i=1}^{M'} \widehat{\omega}_k f(\underline{\boldsymbol{x}}_{i_k}) \tag{1.38}$$

We note that $Q^{CAR}(m,d)$ satisfies the three desired properties (1.24) of a cubature formula

1. $card\big(Q^{CAR}(m,d)\big) = M' \leq M := \binom{m+d}{d}$
2. $\{\underline{\boldsymbol{x}}_{i_k}\}_{k=1}^{M'} \subset \{\underline{\boldsymbol{x}}_i\}_{k=1}^{N} \subset [0,1]^d$
3. $\widehat{\omega}_k \geq 0,\ \forall k \in [\![1, M']\!]$

To ensure that the resulting cubature formula is interpolatory we will need to verify that the weights $\{\widehat{\omega}_k\}_{k=1}^{M'}$ are the unique solution to the system of moment matching equations, formed with nodes $\{\underline{\boldsymbol{x}}_{i_k}\}_{k=1}^{M'}$. As noted in Section 1.2, this is equivalent to requiring that the linear functionals

$$\underline{\boldsymbol{x}}^*_{i_1}, \ldots, \underline{\boldsymbol{x}}^*_{i_{M'}} : \Pi_m^d \to \mathbb{R} \qquad \text{defined by} \quad \underline{\boldsymbol{x}}^*_{i_k}(P) = P(\underline{\boldsymbol{x}}_{i_k}) \quad k = 1, \ldots, M'$$

are linearly independent over Π_m^d.

We can ensure that the latter condition is met as follows. We require that the (m,d)-monomial map $\gamma_m^d := (\gamma_1, \ldots, \gamma_M) : [0,1]^d \to \mathbb{R}^M$ evaluated at $\{\underline{\boldsymbol{x}}_{i_k}\}_{k=1}^{M'}$, the cubature nodes of $Q^{CAR}(m,d)$, forms a full-rank matrix of rank M'. In other words the matrix

$$A' = [\gamma_m^d(\underline{\boldsymbol{x}}_{i_1}), \cdots, \gamma_m^d(\underline{\boldsymbol{x}}_{i_{M'}})] \qquad \text{has} \quad rank(A') = M'$$

1.3.2 Carathéodory Cubature with Simplex problem formulation

Let us now, describe one constructive procedure for finding the collection of weights $\left\{\widehat{\omega}_k\right\}_{k=1}^{M'} \subset \mathbb{R}_+$, satisfying (1.37).

We note that the vector of product cubature weights $\underline{\boldsymbol{\omega}} = [\omega_1, \ldots, \omega_N]^T \in \mathbb{R}_+^N$, arising from the cubature product measure in (1.35), is known as a feasible solution to the following LP(Linear Programming) problem

$$A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}} \tag{1.39}$$

$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}$$

where $A \in \mathbb{R}^{M \times N}, \boldsymbol{b} \in \mathbb{R}^N$ are defined in (1.36). The solution $\underline{\boldsymbol{\omega}}$ is feasible since it satisfies all the constraints of (1.39), and hence lies in the feasible region of (1.39), defined as

$$\mathcal{X} := \left\{\underline{\boldsymbol{x}} \in \mathbb{R}^N \ : \ A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}, \ \ \underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}\right\}$$

Therefore the LP problem (1.39) is said to be feasible, since a feasible solution $\underline{\boldsymbol{\omega}}$ exists. Moreover, we note that $\underline{\boldsymbol{\omega}}$ is a strictly feasible solution since $\underline{\boldsymbol{\omega}} > \underline{\mathbf{0}}$. From [49], we know that every feasible LP problem admits a basic feasible solution. Essentially, a basic feasible solution $\widehat{\underline{\boldsymbol{\omega}}}$ for (1.39) is a solution that is guaranteed to have at least $(N - M)$ zero coordinates, that is, it is given by $\widehat{\underline{\boldsymbol{\omega}}} = [\widehat{\omega}_1, \ldots, \widehat{\omega}_{M'}, 0, \ldots, 0]^T$ with $M' \leq M$. In the following, we provide a formal definition of a basic feasible solution to (1.39) and illustrate how it can be utilised to construct the interpolatory cubature $Q^{CAR}(m, d)$ from $Q(m, d)$, discussed in the previous section.

For simplicity of exposition, let us assume for the moment that $rank(A) = M$. We let $\mathcal{I}_{\mathcal{B}} \triangleq \{j_1, \ldots, j_M\} \subset \{1, \ldots, N\}$ be some subset of indices of the columns of $A = \left\{\underline{\boldsymbol{a}}_j\right\}_{j=1}^N$ that form a linearly independent set, so that

$$B = [\underline{\boldsymbol{a}}_{j_1}, \ldots, \underline{\boldsymbol{a}}_{j_M}] \qquad \text{with} \quad rank(B) = M$$

and let $\mathcal{I}_{\mathcal{N}} \triangleq \{1, \ldots, N\} \backslash \mathcal{I}_{\mathcal{B}}$ be the set of indices of the columns of A, that are not in $\mathcal{I}_{\mathcal{B}}$. We define $B \in \mathbb{R}^{M \times M}$ and $N \in \mathbb{R}^{M \times (N-M)}$, such that $A = [B, N]$. Then, since B is non-singular, we can define a vector $\widehat{\underline{\boldsymbol{\omega}}} = [\ \widehat{\underline{\boldsymbol{\omega}}}_B, \widehat{\underline{\boldsymbol{\omega}}}_N\ ]^T$, given by

$$\widehat{\underline{\boldsymbol{\omega}}}_B = B^{-1}\underline{\boldsymbol{b}} \tag{1.40}$$

$$\widehat{\underline{\boldsymbol{\omega}}}_N = \underline{\mathbf{0}}$$

Then, the vector $\widehat{\underline{\boldsymbol{\omega}}} = [\ \widehat{\underline{\boldsymbol{\omega}}}_B, \widehat{\underline{\boldsymbol{\omega}}}_N\ ]^T = [\widehat{\omega}_1, \ldots, \widehat{\omega}_M, 0, \ldots, 0]^T \in \mathbb{R}^N$ is called a **basic solution**. If $\widehat{\underline{\boldsymbol{\omega}}}_B \geq \underline{\mathbf{0}}$, then $\widehat{\underline{\boldsymbol{\omega}}}$ is called a **basic feasible solution** and if at least one of the components of $\widehat{\underline{\boldsymbol{\omega}}}_B$ is zero, then $\widehat{\underline{\boldsymbol{\omega}}}$ is called a degenerate basic feasible solution. Degeneracy occurs in scenarios, such as when the underlying matrix A is rank deficient.

Relaxing the assumption on the rank of A, that is assuming $rank(A) = M^{'} \leq M$, we let $\mathcal{I}_B \triangleq \{j_1, \ldots, j_{M'}\} \subset \{1, \ldots, N\}$, be a set of indices, such that $\left\{\underline{\boldsymbol{a}}_{j_1}, \ldots, \underline{\boldsymbol{a}}_{j_{M'}}\right\}$ forms a linearly independent set and $\mathcal{I}_N \triangleq \{1, \ldots, N\}/\mathcal{I}_B$. We can then, partition the underlying matrix A as

$$A = [B, N] \quad B \in \mathbb{R}^{M \times M^{'}},\ N \in \mathbb{R}^{M \times (N - M^{'})}$$

where B consists of the subset of columns of A indexed by $\mathcal{I}_B$ and N consists of the subset of columns of A indexed by $\mathcal{I}_N$.

We define a basic feasible solution to (1.39), as a vector

$$\widehat{\underline{\boldsymbol{\omega}}} = [\ \widehat{\underline{\boldsymbol{\omega}}}_B, \widehat{\underline{\boldsymbol{\omega}}}_N\ ]^T = [\widehat{\omega}_1, \ldots, \widehat{\omega}_{M'}, 0, \ldots, 0]^T \in \mathbb{R}^N_+ \tag{1.41}$$

satisfying

$$A\widehat{\underline{\boldsymbol{\omega}}} = B\widehat{\underline{\boldsymbol{\omega}}}_B + N\widehat{\underline{\boldsymbol{\omega}}}_N = \underline{\boldsymbol{b}} + \underline{\mathbf{0}} = \underline{\boldsymbol{b}}, \quad \widehat{\underline{\boldsymbol{\omega}}}_B > \underline{\mathbf{0}},\ \widehat{\underline{\boldsymbol{\omega}}}_N = \underline{\mathbf{0}}$$

See e.g [49], [50]. This is the most general form of the basic feasible solution, that we will be dealing with in the context of this thesis. We recall that the underlying coefficient matrix A in (1.39), is the moment matrix associated with a prescribed collection of cubature nodes $\{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_N\}$, arising from the product cubature $Q(m, d)$ and basis elements $\mathcal{B}(\Pi^d_m) = \{\gamma_1, \ldots, \gamma_M\}$. Thus, for any given degree m and dimension d, we may only conclude that $rank(A) = M^{'} \leq M$ and hence we denote any basic feasible solution to (1.39) by (1.41).

By definition any basic feasible solution (1.41) yields a collection of $M^{'} \leq M$ strictly positive weights $\left\{\widehat{\omega}_k\right\}_{k=1}^{M^{'}}$. We know from the previous section that any such collection of positive weights $\left\{\widehat{\omega}_k\right\}_{k=1}^{M^{'}}$ can be utilised to select $\left\{\underline{\boldsymbol{x}}_{i_k}\right\}_{k=1}^{M^{'}}$ of cubature nodes from the original collection of $\{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_N\}$ cubature nodes arising from the product cubature $Q(m, d)$. Thus, any basic feasible solution (1.41) satisfies the goal of our

construction of Carathéodory cubature $Q^{CAR}(m,d)$, given in (1.38) from the product cubature $Q(m,d)$. Naturally, by the strict positivity of weights in (1.41) and their cardinality $M^{'} \leq M$, the resulting cubature formula $Q^{CAR}(m,d)$ satisfies the three desired properties of a cubature formula, noted in the previous section.

We conclude that appealing to Carathéodory Convex Hull Theorem for the interpolatory cubature problem formulation coupled with the construction of a basic feasible solution to (1.39) given by (1.41), does indeed provide a constructive method for obtaining an interpolatory cubature formula $Q^{CAR}(m,d)$ from $Q(m,d)$.

We are now in a position to readily show that $Q^{CAR}(m,d)$ is indeed an interpolatory cubature formula. That is, we can show that any basic feasible solution (1.41) to (1.39) is unique for the chosen set of abscissae $\left\{\underline{\boldsymbol{x}}_{i_k}\right\}_{k=1}^{M^{'}}$. As noted at the end of the previous section, this is equivalent to showing that the set of columns of A corresponding to the strictly positive weights $\left\{\widehat{\omega}_k\right\}_{k=1}^{M^{'}}$ is linearly independent in $\mathbb{R}^M$. By construction of the basic feasible solution, the set of columns of A corresponding to the strictly positive weights in (1.41) is given by $\{\underline{\boldsymbol{a}}_{i_k}\}_{k=1}^{M^{'}} = \left\{\gamma_m^d(\underline{\boldsymbol{x}}_{i_k})\right\}_{k=1}^{M^{'}}$ and is linearly independent subset of $\mathbb{R}^M$. Hence, defining $A^{'} = [\gamma_m^d(\underline{\boldsymbol{x}}_{i_1}), \cdots, \gamma_m^d(\underline{\boldsymbol{x}}_{i_{M^{'}}})]$, we have $rank(A^{'}) = M^{'}$, as required.

In the aforementioned construction of $Q^{CAR}(m,d)$ from a given product cubature $Q(m,d)$, we state that any basic feasible solution to (1.39) of the form (1.41) will yield the desired interpolatory cubature $Q^{CAR}(m,d)$. Thus, we allude to the non-uniqueness of basic feasible solutions to (1.39). Naturally, this means that $Q^{CAR}(m,d)$ is also not unique, however for the purposes of this work, we do not differentiate between different basic feasible solutions (1.41) to (1.39), as they all yield an interpolatory cubature formula satisfying the three desired cubature properties.

To illustrate the existence of multiple basic feasible solutions to (1.39), in the following we present a geometric interpretation of the LP problem (1.39).

Geometrically, finding a basic feasible solution $\underline{\widehat{\boldsymbol{\omega}}}$ to (1.39), amounts to finding an extreme point of the feasible region

$$\mathcal{X} := \left\{\underline{\boldsymbol{x}} \in \mathbb{R}^N \ : \ A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}, \ \ \underline{\boldsymbol{x}} \geq \underline{\boldsymbol{0}}\right\} \tag{1.42}$$

To introduce the notion of extreme points of the feasible region $\mathcal{X}$, we recall the following definitions.

A set $H \subseteq \mathbb{R}^N$ is called a hyperplane if $H = \{\underline{\boldsymbol{x}} \in \mathbb{R}^N \ : \ \underline{\boldsymbol{a}}^T\underline{\boldsymbol{x}} = c\}$ for some non-zero $\underline{\boldsymbol{a}} \in \mathbb{R}^N$ and $c \in \mathbb{R}$ and similarly $H^{'} \subseteq \mathbb{R}^N$ is called a half-space if $H^{'} = \{\underline{\boldsymbol{x}} \in \mathbb{R}^N \ : \ \underline{\boldsymbol{a}}^T\underline{\boldsymbol{x}} \geq c\}$. Both hyperplanes and half-spaces are convex sets, that is sets in which any two points $\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_2 \in H$ have their convex combination $\lambda\underline{\boldsymbol{x}}_1 + (1-\lambda)\underline{\boldsymbol{x}}_2 \in H$ for every $\lambda \in [0,1]$. Since the feasible region $\mathcal{X}$ is a finite intersection of hyperplanes and half-spaces and finite intersection of convex sets is convex, we know that $\mathcal{X}$ is convex.

A point $\underline{\boldsymbol{x}}$ in the convex set $\mathcal{X}$ is called an extreme point if $\underline{\boldsymbol{x}}$ cannot be represented as a strict convex combination of the two distinct points in $\mathcal{X}$, that is if $\underline{\boldsymbol{x}} = \lambda\underline{\boldsymbol{x}}_1 + (1-\lambda)\underline{\boldsymbol{x}}_2$ with $\lambda \in (0,1)$ and $\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_2 \in \mathcal{X}$, then we must have $\underline{\boldsymbol{x}} = \underline{\boldsymbol{x}}_1 = \underline{\boldsymbol{x}}_2$. An extreme point of $\mathcal{X}$ is a point in $\mathcal{X}$ that cannot be made to lie in the interior of a line segment contained in $\mathcal{X}$, and hence is also known as a corner point or vertex.

The following lemma illustrates the correspondence between basic feasible solutions and extreme points of the feasible region.

Lemma 1.3.2. *[[51] Lemma 7.2] Given an LP problem in the standard form*

$$\begin{aligned} \textit{Maximise} \quad & \underline{\boldsymbol{c}}^T\underline{\boldsymbol{x}} \\ \textit{subject to} \quad & A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}} \\ & \underline{\boldsymbol{x}} \geq \underline{\mathbf{0}} \end{aligned} \tag{1.43}$$

where $A \in \mathbb{R}^{M\times N}$, $\underline{\boldsymbol{x}} \in \mathbb{R}^N$, $\underline{\boldsymbol{b}} \in \mathbb{R}^M$, suppose that $rank(A) = M$ and the columns of A are ordered such that $A = [B, N]$, where $B \in \mathbb{R}^{M\times M}$ is non-singular and $N \in \mathbb{R}^{M\times(N-M)}$. Then, the basic feasible solution $\widehat{\underline{\boldsymbol{\omega}}} = [\ \widehat{\underline{\boldsymbol{\omega}}}_B, \widehat{\underline{\boldsymbol{\omega}}}_N\ ]^T$, given by

$$\widehat{\underline{\boldsymbol{\omega}}}_B = B^{-1}\underline{\boldsymbol{b}}$$

$$\widehat{\underline{\boldsymbol{\omega}}}_N = \underline{\mathbf{0}}$$

is an extreme point of the feasible region of (1.43).

Hence, we know that if the underlying coefficient matrix A is full-ranked, the collection of extreme points of the feasible region corresponds to the collection of basic

feasible solutions, provided that the feasible region is non empty. We note that any basic feasible solution $\widehat{\underline{\boldsymbol{\omega}}}$ to (1.43) is given by identifying $\mathcal{I}_{\mathcal{B}} = \{j_1, \ldots, j_M\} \subset \{1, \ldots, N\}$, be some subset of indices of the columns of A that form a linearly independent set, thus giving an upper bound of $\binom{N}{M}$ on the number of possible basic feasible solutions or extreme points of the feasible region.

In practice, we will endeavour to find only one basic feasible solution to (1.39), as we do not differentiate as to the quality of basic feasible solutions to yield a Carathéodory cubature measure in a construction of (1.38). In the following, we proceed to illustrate how we can find one basic feasible solution to (1.39).

In the Simplex problem formulation (1.39), firstly we assume that the underlying matrix A has no identity submatrix, as otherwise an immediate basic feasible solution is available to us by setting $B = I$, yielding $\widehat{\underline{\boldsymbol{\omega}}} = [\, \widehat{\underline{\boldsymbol{\omega}}}_B, \widehat{\underline{\boldsymbol{\omega}}}_N \,]^T$ with $\widehat{\underline{\boldsymbol{\omega}}}_B = B^{-1}\underline{\boldsymbol{b}} = \underline{\boldsymbol{b}} \geq \underline{\mathbf{0}}$.

A standard method for finding an initial basic feasible solution to (1.39) is to introduce a vector of artificial variables $\underline{\boldsymbol{x}}_a$ and thus get a basic feasible solution to a modified problem. Subsequently, the Simplex method can be employed to force these artificial variables to zero and obtain a basic feasible solution to the original problem (1.39). For a detailed discussion see Chapter 4 [50] or Chapter 8.4 [51].

We proceed by adding a non-negative vector of artificial variables $\underline{\boldsymbol{x}}_a \in \mathbb{R}^M_+$ to the constraints in (1.39), resulting in the following formulation

$$A\underline{\boldsymbol{x}} + \underline{\boldsymbol{x}}_a = \underline{\boldsymbol{b}} \tag{1.44}$$

$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}},\ \underline{\boldsymbol{x}}_a \geq \underline{\mathbf{0}}$$

By construction of (1.44), we introduce an identity matrix corresponding to $\underline{\boldsymbol{x}}_a$. Thus a basic feasible solution to the modified problem (1.44) is readily available, namely

$$[\underline{\boldsymbol{x}},\ \underline{\boldsymbol{x}}_a] = [\underline{\mathbf{0}},\ \underline{\boldsymbol{b}}]$$

However, in order to get back to our original problem, we must force the artificial variables to zero, since $A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}$ if and only if $A\underline{\boldsymbol{x}} + \underline{\boldsymbol{x}}_a = \underline{\boldsymbol{b}}$ with $\underline{\boldsymbol{x}}_a = \underline{\mathbf{0}}$. A widely employed method to accomplish this, see Chapter 4 [50] or Chapter 8.4 [51], is to

minimise the sum of the artificial variables, which amounts to solving the following LP problem

$$\begin{aligned} \text{Minimise} \quad & \sum_{i=1}^{M} x_a^i \\ \text{subject to} \quad & A\underline{\boldsymbol{x}} + \underline{\boldsymbol{x}}_a = \underline{\boldsymbol{b}} \\ & \underline{\boldsymbol{x}} \geq \underline{\mathbf{0}},\ \underline{\boldsymbol{x}}_a \geq \underline{\mathbf{0}} \end{aligned} \tag{1.45}$$

We solve (1.45) by the Simplex Method or Interior Point method described at the end of this chapter, starting with initial basic feasible solution $[\underline{\boldsymbol{x}},\ \underline{\boldsymbol{x}}_a] = [\underline{\mathbf{0}},\ \underline{\boldsymbol{b}}]$. If the original problem (1.39) has a feasible solution, then the optimal value of (1.45) is zero. Moreover, as the artificial variables x_a^i leave the basis, they are replaced with legitimate variables and we obtain a basic feasible solution to the original problem (1.39).

An approach very similar to the aforementioned method, utilising basic feasible solution to (1.39) for constructing $Q^{CAR}(m, d)$ is presented in [48]. In [48], the authors subdivide their for producing interpolatory method cubature into two categories: an independent case, in which the underlying density $\rho(\underline{\boldsymbol{x}})$ is factorable into a product of univariate densities, that is

$$\rho(\underline{\boldsymbol{x}}) = \rho(x^1, \ldots, x^d) = \prod_{i=1}^{d} \rho_i(x^i) \quad \text{with } \rho_i$$

and the jointly distributed case, in which the density $\rho(\underline{\boldsymbol{x}})$ is not factorable.

Implicitly, in our construction of the Cartesian product formulae in (1.35) we have chosen the d-dimensional Lebesgue density $\rho(\underline{\boldsymbol{x}}) = 1$, which clearly falls into the independent category in [48].

In Chapter 4 of this dissertation we introduce a more general framework for the Carathéodory cubature algorithm that deals with any factorable density for which univariate quadrature formulae are available. This generalised algorithm presented in Chapter 4 will also fall into the independent case category. In order to deal with other case, the case of joint densities, that are not factorable the authors in [48] proceed by setting up a grid over the range of each marginal variable and construct a Cartesian

product of these grids to obtain a lattice in d dimensions and subsequently proceed to employ the Simplex problem formulation resulting from the cubature on the lattice to produce an interpolatory cubature. However, they note that due to such a heuristic selection of initial cubature points and weight there is no guarantee that the resulting linear program is feasible. In addition, it is noted that increasing the density of the underlying mesh should restore feasibility and thus yield an interpolatory cubature formula. In the present work, we note that an approach similar to [48] can certainly be employed to deal with non-factorable densities within the framework of Carathéodory cubatures, however it is outside the scope of this work to investigate such a method and its potential fallibility in yielding a basic feasible solution and thus an interpolatory cubature.

1.3.2.1 Computational Complexity of the Carathéodory Cubature with Simplex problem formulation

The computational complexity of constructing $Q^{CAR}(m,d)$ via the Simplex problem formulation, is dominated by the construction of the Cartesian product cubature formula in (1.35), which requires touching N cubature nodes of dimension d and N weights, thus having a cost of $N(d+1)$, constructing the linear system (1.39), which has an associated cost of NM and computing a basic feasible solution to (1.39), which has an associated cost of $C_{SIMPL}(M,N)$.

Thus the overall cost of constructing $Q^{CAR}(m,d)$ via the Simplex problem formulation is dominated by

$$C\Big(Q^{CAR}(m,d)\Big) = O\Big(NM + C_{SIMPL}(M,N)\Big)$$

As we shall illustrate in Section 1.4, the computational complexity of finding a basic feasible solution to (1.39) via the Simplex Algorithm, is on average given by

$$C_{SIMPL}(M,N) = O(M^3N + MN^2)$$

Yielding the overall cost of computing $Q^{CAR}(m,d)$ to be of order

$$C\Big(Q^{CAR}(m,d)\Big) = O(NM + M^3N + MN^2)$$

The cost $O(NM)$ associated with the initial Cartesian product cubature construction and constructing the linear system (1.39), can reduced via a recursive algorithm to $O(M^2)$, independent of N, as will be illustrated in Chapter 4. Without explicit proof, we conclude that the theoretical cost of constructing $Q^{CAR}(m,d)$ with Simplex problem formulation is given by

$$C\Big(Q^{CAR}(m,d)\Big) = O(M^3N + MN^2)$$

we note that this cost has a quadratic dependence on N. In the vast majority of problems, we will have $N >> M$, where $N = card(Q(m,d)) = n^d$ is the number of cubature modes in the Cartesian product cubature $Q(m,d)$, which grows exponentially with dimension. Therefore, the complexity of the Carathéodory Cubature construction with Simplex problem formulation, grows exponentially with dimension and thus becomes intractable for sufficiently large dimensions d.

1.3.3 Carathéodory cubature with SVD problem fomulation

In the following, we shall present an alternative problem formulation for computing Carathéodory cubature that does not rely on utilising the Simplex or Interior Point methods to find a basic feasible solution. This new problem formulation will delineate the main contribution of this dissertation: an efficient algorithm for finding a basic feasible solution to (1.39), in the case when a strictly feasible solution is at hand.

The novel algorithm, we are about to describe does not utilise any special structure of the LP problem (1.39) and thus is more widely applicable in the context of computing a basic feasible solution to any LP problem, when a a strictly feasible solution is known. Hence, we shall firstly present the algorithm in the most general framework and subsequently specialise it to (1.39).

1.3.3.1 Formulation of general framework

We recall that any linear programming problem can be transformed into a problem of the following form:

$$\begin{aligned} &\text{Min/Max} \quad \underline{\boldsymbol{c}}^T \underline{\boldsymbol{x}} \\ &\text{subject to} \quad A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}} \\ &\qquad\qquad \underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}, \quad \underline{\boldsymbol{b}} \geq \underline{\mathbf{0}} \end{aligned} \tag{1.46}$$

This is achieved by manipulating constraints and introducing slack variables, for more details see Chapter 4 in [50]. In the following, we assume that $A = [\underline{\boldsymbol{a}}_1, \ldots, \underline{\boldsymbol{a}}_N] \in \mathbb{R}^{M\times N}$, $\underline{\boldsymbol{x}} = [x_1, \ldots, x_N]^T \in \mathbb{R}^N_+$ and $\underline{\boldsymbol{b}} = [b_1, \ldots, b_M]^T \in \mathbb{R}^M_+$.

Let us further assume that a strictly feasible solution $\underline{\boldsymbol{\omega}} = [\omega_1, \ldots, \omega_N]^T \in \mathbb{R}^N_+$ to (1.46) is given, that is

$$A\underline{\boldsymbol{\omega}} = \underline{\boldsymbol{b}}, \quad \underline{\boldsymbol{\omega}} > \underline{\mathbf{0}}$$

Since (1.46) has a feasible solution, the problem is feasible and thus has a basic feasible solution. To construct an initial feasible solution $\widehat{\underline{\boldsymbol{\omega}}} = [\widehat{\omega}_1, \ldots, \widehat{\omega}_M, 0, \ldots, 0] \in \mathbb{R}^N_+$ to (1.46) we can utilise the Simplex Algorithm, as detailed in the previous section.

In the following, we shall present an alternative algorithm for finding a basic feasible solution to (1.46). We consider the following modified system

$$\tilde{A}\underline{\tilde{\boldsymbol{\omega}}} = \underline{\tilde{\boldsymbol{b}}}, \quad \underline{\tilde{\boldsymbol{\omega}}} > \underline{\mathbf{0}}$$

where

$$\tilde{A} = [\underline{\tilde{\boldsymbol{a}}}_1, \ldots, \underline{\tilde{\boldsymbol{a}}}_N] := \begin{bmatrix} 1 & \ldots & 1 \\ \underline{\boldsymbol{a}}_1 & \ldots & \underline{\boldsymbol{a}}_N \end{bmatrix} \in \mathbb{R}^{(M+1)\times N}$$

$$\underline{\tilde{\boldsymbol{\omega}}} = [\tilde{\omega}_1, \ldots, \tilde{\omega}_N] := \left[\frac{\omega_1}{\|\underline{\boldsymbol{\omega}}\|_1}, \ldots, \frac{\omega_1}{\|\underline{\boldsymbol{\omega}}\|_N}\right] \in \mathbb{R}_+^N$$

$$\underline{\tilde{\boldsymbol{b}}} := [1, b_1, \ldots, b_M] \in \mathbb{R}_+^{M+1}$$

We will construct a basic feasible solution $\underline{\widehat{\boldsymbol{\omega}}} = [\widehat{\omega}_1, \ldots, \widehat{\omega}_M, 0, \ldots, 0] \in \mathbb{R}_+^N$, which is simultaneously a basic feasible solution for

$$\begin{aligned} &\text{Min/Max} \quad \underline{\boldsymbol{c}}^T \underline{\boldsymbol{x}} \\ &\text{subject to} \quad \tilde{A}\underline{\boldsymbol{x}} = \underline{\tilde{\boldsymbol{b}}} \\ &\qquad\qquad \underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}, \quad \underline{\tilde{\boldsymbol{b}}} \geq \underline{\mathbf{0}} \end{aligned} \tag{1.47}$$

and for the original LP problem of interest (1.46). This holds, since the LP problem (1.47) has only one additional constraint, compared to (1.46), namely the solution vector is required to add up to one.

We note that, since $\underline{\boldsymbol{\omega}} = [\omega_1, \ldots, \omega_N]^T \in \mathbb{R}_+^N$ is a strictly feasible solution to (1.46), we have that $\underline{\tilde{\boldsymbol{\omega}}} = [\tilde{\omega}_1, \ldots, \tilde{\omega}_N]^T \in \mathbb{R}_+^N$ is a strictly feasible solution to (1.47), that is

$$\tilde{A}\underline{\tilde{\boldsymbol{\omega}}} = \underline{\tilde{\boldsymbol{b}}}, \quad \underline{\tilde{\boldsymbol{\omega}}} > \underline{\mathbf{0}} \tag{1.48}$$

Starting from (1.47), we proceed to formulate a sequence of LP problems of dimensions independent of N, by means of hierarchical clustering of the LP problem (1.47). Eliminating the dependence on the parameter N is often crucial in reducing the complexity of the overall algorithm, as many LP problems assume $N >> M$. Subsequently, we

present a novel approach for finding a basic feasible solution to each of the LP problems generated. We then, proceed to sequentially produce basic feasible solutions to each smaller LP problem in turn and culminate with obtaining a basic feasible solution to the original problem (1.47).

1.3.3.2 Hierarchical Clustering

For the purposes of hierarchical cluster representation, we shall be concerned with the constraint data $\left\{(\tilde{\omega}_i, \underline{\tilde{\boldsymbol{a}}}_i)\right\}_{i=1}^{N}$ of (1.48). For simplicity of exposition, we shall assume that $N = 2^S$ and we have some $M = 2^{R-1}$, with $S > R$. We define $D \triangleq \log_2\left(\frac{N}{2M}\right)$. With these assumptions we can construct the coarsest hierarchical level of $2M$ clusters $\left\{C_1, \ldots, C_{2M}\right\}$, satisfying

$$C_i \subset \left\{(\tilde{\omega}_i, \underline{\tilde{\boldsymbol{a}}}_i)\right\}_{i=1}^{N}, \quad C_i \cap C_\ell = \emptyset \quad \text{for } i \neq \ell, \quad \bigcup_{i=1}^{2M} C_i = \left\{(\tilde{\omega}_i, \underline{\tilde{\boldsymbol{a}}}_i)\right\}_{i=1}^{N}, \ |C_i| = 2^D$$

Each cluster C_i will be represented by its centroid $(\lambda_i, \underline{\tilde{\boldsymbol{y}}}_i)$, where

$$\lambda_i = \sum_{j:\tilde{\omega}_j \in C_i} \tilde{\omega}_j, \quad \underline{\tilde{\boldsymbol{y}}}_i = \frac{1}{\lambda_i}\left(\sum_{\underline{\tilde{\boldsymbol{a}}}_j \in C_i} \tilde{\omega}_j \underline{\tilde{\boldsymbol{a}}}_j\right)$$

Moreover, each cluster C_i can be hierarchically subdivided into a collection of subclusters as follows

$$C_i = C_{2i-1}^{D-1} \cup C_{2i}^{D-1} = C_{4i-3}^{D-2} \cup C_{4i-2}^{D-2} \cup C_{4i-1}^{D-2} \cup C_{4i}^{D-2} = \ldots$$

$$= C_{(i-1)2^D+1}^{0} \cup \ldots \cup C_{i2^D}^{0}$$

where every subcluster C_k^j satisfies

$$C_k^j \subset C_i, \ C_k^j = C_{2k-1}^{j-1} \cup C_{2k}^{j-1}, \quad |C_k^j| = \frac{N}{2^j}, \ |C_{2k-1}^{j-1}| = |C_{2k}^{j-1}| = \frac{N}{2^{j-1}},$$

The hierarchical cluster representation is constructed using an agglomerative clustering paradigm, starting with the collection $\left\{(\tilde{\omega}_i, \underline{\tilde{\boldsymbol{a}}}_i)\right\}_{i=1}^{N}$ as singleton clusters, we

recursively merge each adjacent pair of clusters into a single subcluster, until we have formed the desired $2M$ clusters $\{C_1, \ldots, C_{2M}\}$. We proceed as follows

Set $C_i^0 \triangleq \{(\tilde{\omega}_i, \underline{\tilde{\boldsymbol{a}}}_i)\}$ for $k \in [\![1, N]\!]$

with centroids $(\lambda_i^0, \underline{\tilde{\boldsymbol{y}}}_i^0) \triangleq (\tilde{\omega}_i, \underline{\tilde{\boldsymbol{a}}}_i)$

For $j = 1 : D,\ k = 1 : 2^{D-j}$

$C_k^j \triangleq C_{2k-1}^{j-1} \cup C_{2k}^{j-1}$

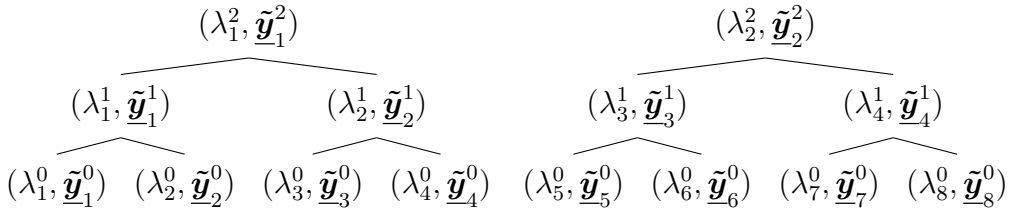
with centroids $(\lambda_k^j, \underline{\tilde{\boldsymbol{y}}}_k^j) \triangleq \left(\lambda_{2k-1}^{j-1} + \lambda_{2k}^{j-1}, \dfrac{\lambda_{2k-1}^{j-1} \underline{\tilde{\boldsymbol{y}}}_{2k-1}^{j-1} + \lambda_{2k}^{j-1} \underline{\tilde{\boldsymbol{y}}}_{2k}^{j-1}}{\lambda_{2k-1}^{j-1} + \lambda_{2k}^{j-1}} \right)$

Output $C_i \triangleq C_i^D$ for $i \in [\![1, 2M]\!]$

with centroids $(\lambda_i^D, \underline{\tilde{\boldsymbol{y}}}_i^D)$

This recursive agglomeration of clusters has a natural representation as a binary forest of depth D. Each tree in the forest represents a cluster C_i with all its constituent subclusters $C_k^j \subset C_i$. The nodes of each tree are labelled by the corresponding centroids $(\lambda_k^j, \underline{\tilde{\boldsymbol{y}}}_k^j)$ of C_k^j. Root nodes of the forest $(\lambda_k^D, \underline{\tilde{\boldsymbol{y}}}_k^D)$ are the centroids of the coarsest clustering level $\{C_1, \ldots, C_{2M}\}$ and leaf nodes $(\lambda_k^0, \underline{\tilde{\boldsymbol{y}}}_k^0)$ represent the original singleton clusters $\{(\tilde{\omega}_1, \underline{\tilde{\boldsymbol{a}}}_1), \ldots, (\tilde{\omega}_N, \underline{\tilde{\boldsymbol{a}}}_N)\}$.

In the following, we illustrate the binary forest representation for $N = 2^3$ and $M = 1$.



By (1.48), we note that the centre of mass of the centroids for each prescribed hierarchical level $j \in [\![0, D]\!]$ satisfies

$$\sum_{j=1}^{2^{S-j}} \lambda_k^j \underline{\tilde{\boldsymbol{y}}}_k^j = \underline{\tilde{\boldsymbol{b}}} \tag{1.49}$$

In the following, we illustrate how the hierarchical cluster representation can be utilised to construct a smaller LP problem from (1.47) on the clusters of original constraint data and present a novel approach for computing a basic feasible solution to the LP problem generated.

1.3.3.3 SVD Algorithm for Basic Feasible Solution Construction

Let a collection of clusters $\{C_1, \ldots, C_{2M}\}$ with centroids $\{(\lambda_1, \underline{\tilde{\boldsymbol{y}}}_1), \ldots, (\lambda_{2M}, \underline{\tilde{\boldsymbol{y}}}_{2M})\}$ and centre of mass of centroids $\sum_{j=1}^{2M} \lambda_k \underline{\tilde{\boldsymbol{y}}}_k = \underline{\tilde{\boldsymbol{b}}}$ be given. Set

$$C = C^{(0)} \triangleq [\underline{\tilde{\boldsymbol{y}}}_1, \ldots, \underline{\tilde{\boldsymbol{y}}}_{2M}] \in \mathbb{R}^{(M+1)\times 2M}$$

$$\underline{\boldsymbol{\lambda}} = \underline{\boldsymbol{\lambda}}^{(0)} \triangleq [\lambda_1, \ldots, \lambda_{2M}] \in \mathbb{R}_+^{2M}$$

Construct the following feasibility LP problem (with trivial objective function)

$$C\underline{\boldsymbol{x}} = \underline{\tilde{\boldsymbol{b}}} \tag{1.50}$$

$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}$$

By (1.49), we know that $\underline{\boldsymbol{\lambda}}$ is a strictly feasible solution to (1.50), since

$$C\underline{\boldsymbol{\lambda}} = \underline{\tilde{\boldsymbol{b}}}, \quad \underline{\boldsymbol{\lambda}} > \underline{\mathbf{0}}$$

Therefore since (1.50) is feasible, a basic feasible solution exists and can be obtained via the Simplex algorithm. The cost of this computation is on average of order $O(M^4 + M^3)$. In the following, we present an alternative Singular Value Decomposition based algorithm for constructing a basic feasible solution to (1.50) with a computational cost of order $O(M^3)$.

For simplicity of exposition, we assume that $rank(C^{(0)}) = M$. We proceed to compute the Singular Value Decomposition of $C^{(0)}$, given by

$$C^{(0)} = U\Sigma V^T = \left[\begin{array}{cc} U_M & U_0 \end{array}\right] \left[\begin{array}{c|c} \Sigma_M & 0 \end{array}\right] \left[\begin{array}{c} V_M^T \\ \\ V_0^T \end{array}\right] \tag{1.51}$$

where $U \in \mathbb{R}^{(M+1)\times(M+1)}$, $V \in \mathbb{R}^{2M\times 2M}$ are orthonormal matrices, $\Sigma_M = diag(\sigma_1, \ldots, \sigma_M)$ with $\sigma_1 \geq \cdots \geq \sigma_M > 0$. We recall that the Singular Value Decomposition of $C^{(0)}$ provides orthogonal bases for all four fundamental subspaces associated with $C^{(0)}$. In particular, we can readily obtain an orthogonal basis for the $\mathcal{K}er(C^{(0)})$

$$\mathcal{B}\big(\mathcal{K}er(C^{(0)})\big) = \mathcal{C}ol(V_0)$$

which we shall denote by

$$\Phi^{(0)} \triangleq [\underline{\boldsymbol{\phi}}_1, \ldots, \underline{\boldsymbol{\phi}}_M] = \mathcal{C}ol(V_0) \in \mathbb{R}^{2M\times M}$$

We note that any null vector of $C^{(0)}$ for $1 \leq i \leq M$ satisfies

$$\sum_{j=1}^{2M} (\underline{\boldsymbol{\phi}}_i)_j \underline{\tilde{\boldsymbol{y}}}_j = \underline{\mathbf{0}}$$

and in particular since $\tilde{y}_{1,j} = 1$, $\forall j \in [\![1, 2M]\!]$, we have

$$\sum_{j=1}^{2M} (\underline{\boldsymbol{\phi}}_i)_j = 0$$

Thus, at least one $(\underline{\boldsymbol{\phi}}_i)_j > 0$, as $\underline{\boldsymbol{\phi}}_i \neq \underline{\mathbf{0}}$.

Furthermore, we note that for any $\alpha \in \mathbb{R}$ and any $1 \leq i \leq M$, we have

$$\underline{\tilde{\boldsymbol{b}}} = C^{(0)}\underline{\boldsymbol{\lambda}}^{(0)} = C^{(0)}\underline{\boldsymbol{\lambda}}^{(0)} - \alpha\big(C^{(0)}\underline{\boldsymbol{\phi}}_i\big) \tag{1.52}$$

In particular, we can define

$$\alpha_{(1)} := \min_{1\leq j\leq N}\left\{\frac{\tilde{\omega}_j}{(\underline{\boldsymbol{\phi}}_i)_j} : (\underline{\boldsymbol{\phi}}_i)_j > 0\right\} = \frac{\tilde{\omega}_k}{(\underline{\boldsymbol{\phi}}_i)_k} \quad \text{for some } k \in [\![1, 2M]\!]$$

We make an arbitrary choice of $i = 1$ in the definition of $\alpha_{(1)}$ and in (1.52), which yields the following

$$\underline{\tilde{\boldsymbol{b}}} = C^{(0)}\underline{\boldsymbol{\lambda}}^{(0)} = \sum_{j=1}^{2M} \lambda_j \underline{\tilde{\boldsymbol{y}}}_j - \alpha_{(1)}\left(\sum_{j=1}^{2M} (\underline{\boldsymbol{\phi}}_1)_j \underline{\tilde{\boldsymbol{y}}}_j\right) = \sum_{j=1}^{2M} \big(\lambda_j - \alpha_{(1)}(\underline{\boldsymbol{\phi}}_1)_j\big)\underline{\tilde{\boldsymbol{y}}}_j = \sum_{j=1}^{2M-1} \lambda_j^{(1)} \underline{\tilde{\boldsymbol{y}}}_j$$

where $\underline{\boldsymbol{\lambda}}^{(1)} \triangleq [\lambda_1^{(1)}, \ldots, \lambda_{k-1}^{(1)}, \lambda_{k+1}^{(1)}, \ldots, \lambda_{2M}^{(1)}]^T \in \mathbb{R}_+^{2M-1}$ is given by

$$\lambda_j^{(1)} = \begin{cases} \lambda_j - \alpha_{(1)}(\underline{\boldsymbol{\phi}}_1)_j & \text{for } j \in [\![1, k-1]\!] \\ \lambda_{j+1} - \alpha_{(1)}(\underline{\boldsymbol{\phi}}_1)_{j+1} & \text{for } j \in [\![k, 2M-1]\!] \end{cases}$$

and

$$\sum_{j=1}^{2M-1} \lambda_j^{(1)} = 1$$

Thus, since $\lambda_k - \alpha_{(1)}(\underline{\boldsymbol{\phi}}_1)_k = 0$, we have also produced a new feasible solution to (1.50),

$$[\lambda_1^{(1)}, \ldots, \lambda_{k-1}^{(1)}, 0, \lambda_{k+1}^{(1)}, \ldots, \lambda_{2M}^{(1)}]^T \in \mathbb{R}_+^{2M}$$

placing zero mass on the k^{th} coordinate $\underline{\tilde{\boldsymbol{y}}}_k$.

We proceed to form a new matrix

$$C^{(1)} = [\underline{\tilde{\boldsymbol{y}}}_1, \ldots, \underline{\tilde{\boldsymbol{y}}}_{k-1}, \underline{\tilde{\boldsymbol{y}}}_{k+1}, \ldots, \underline{\tilde{\boldsymbol{y}}}_{2M}] \in \mathbb{R}^{(M+1)\times(2M-1)}$$

and update $\Phi^{(0)} = \mathcal{B}(\mathcal{K}er(C^{(1)}))$ to obtain $\Phi^{(1)} = \mathcal{B}(\mathcal{K}er(C^{(1)}))$, as follows

$$\begin{aligned}
&1)\ \Phi^{(0)} = [\underline{\boldsymbol{\phi}}_1, \ldots, \underline{\boldsymbol{\phi}}_M] \rightarrow \tilde{\Phi}^{(0)} = [\underline{\boldsymbol{\phi}}_2, \ldots, \underline{\boldsymbol{\phi}}_M] \\
&2)\ \tilde{\Phi}^{(0)} = [\underline{\boldsymbol{\phi}}_2, \ldots, \underline{\boldsymbol{\phi}}_M] \rightarrow \Phi^{(1)} = [\widehat{\underline{\boldsymbol{\phi}}}_2, \ldots, \widehat{\underline{\boldsymbol{\phi}}}_M] \\
&\text{where} \quad \widehat{\underline{\boldsymbol{\phi}}}_j = \underline{\boldsymbol{\phi}}_{j+1} - \Big[\frac{(\underline{\boldsymbol{\phi}}_{j+1})_k}{(\underline{\boldsymbol{\phi}}_1)_k}\Big]\underline{\boldsymbol{\phi}}_1
\end{aligned}$$

Analogously, we can iterate the aforementioned procedure for $1 \leq i \leq M$, with

$$C^{(i-1)} \in \mathbb{R}^{(M+1)\times(2M-(i-1))} \rightarrow C^{(i)} \in \mathbb{R}^{(M+1)\times(2M-i)}$$

$$\underline{\boldsymbol{\lambda}}^{(i-1)} \in \mathbb{R}_+^{(2M-(i-1))} \rightarrow \underline{\boldsymbol{\lambda}}^{(i)} \in \mathbb{R}_+^{(2M-i)}$$

$$\Phi^{(i-1)} \in \mathbb{R}^{(2M-(i-1))\times(M-i)} \rightarrow \Phi^{(i)} \in \mathbb{R}^{(2M-i)\times(M-i)}$$

until we have no more null vectors left at iteration $i = M$. Thus, we produce

$$C^{(M)} = [\underline{\tilde{\boldsymbol{y}}}_{i_1}, \ldots, \underline{\tilde{\boldsymbol{y}}}_{i_M}] \in \mathbb{R}^{(M+1)\times M}$$

$$\underline{\boldsymbol{\lambda}}^{(M)} = [\lambda_1^{(M)}, \ldots, \lambda_M^{(M)}] \in \mathbb{R}_+^M$$

satisfying

$$C^{(M)}\underline{\boldsymbol{\lambda}}^{(M)} = \underline{\tilde{\boldsymbol{b}}}, \quad \underline{\boldsymbol{\lambda}}^{(M)} > \underline{\mathbf{0}}$$

Letting $\mathcal{I}_B \triangleq \{i_1, \ldots, i_M\} \subset \{1, \ldots, 2M\}$ and $\mathcal{I}_N \triangleq \{1, \ldots, 2M\}/\mathcal{I}_B$, we can partition the underlying constraint matrix $C = [\underline{\tilde{\boldsymbol{y}}}_1, \ldots, \underline{\tilde{\boldsymbol{y}}}_{2M}]$ in (1.50) as

$$C = [B, N] \quad B \in \mathbb{R}^{(M+1)\times M},\ N \in \mathbb{R}^{(M+1)\times M}$$

where $B \triangleq C^{(M)} = [\underline{\tilde{\boldsymbol{y}}}_{i_1}, \ldots, \underline{\tilde{\boldsymbol{y}}}_{i_M}]$ consists of the subset of columns of C indexed by $\mathcal{I}_B$, forming a linearly independent set, and N consists of the subset of columns of C indexed by $\mathcal{I}_N$.

We can construct a basic feasible solution to (1.50) as

$$\underline{\widehat{\boldsymbol{\lambda}}} = [\underline{\widehat{\boldsymbol{\lambda}}}_B, \underline{\widehat{\boldsymbol{\lambda}}}_N]^T = [\widehat{\lambda}_1, \ldots, \widehat{\lambda}_M, 0, \ldots, 0] \triangleq [\lambda_1^{(M)}, \ldots, \lambda_M^{(M)}, 0, \ldots, 0] \in \mathbb{R}_+^{2M}$$

which satisfies

$$C\underline{\widehat{\boldsymbol{\lambda}}} = B\underline{\widehat{\boldsymbol{\lambda}}}_B + N\underline{\widehat{\boldsymbol{\lambda}}}_N = \underline{\tilde{\boldsymbol{b}}} + \underline{\mathbf{0}} = \underline{\tilde{\boldsymbol{b}}}, \quad \underline{\widehat{\boldsymbol{\lambda}}}_B > \underline{\mathbf{0}},\ \underline{\widehat{\boldsymbol{\lambda}}}_N = \underline{\mathbf{0}}$$

We note that the computational complexity of this approach to finding a basic feasible solution is dominated by the computation of $\mathcal{K}er(C^{(0)})$ and the kernel updates $\Phi^{(i-1)} \rightarrow \Phi^{(i)}$.

The kernel computation via the Singular Value Decomposition of an $(M+1) \times 2M$ matrix is a computation of order $O(M^3)$. Whilst each of the kernel updates are equivalent to subtracting two matrices of dimensions $(2M - (i-1)) \times (M-i)$. Hence, since the kernel update is performed M times, the associated computational cost is $M \times (2M - (i-1)) \times (M-i) = O(M^3)$. Therefore, clearly the overall cost of the algorithm is of order $O(M^3)$.

1.3.3.4 Recombination Algorithm

Given the hierarchical cluster representation of the constraint data in (1.48)

$$\left\{C_1 \cup \ldots, \cup C_{2M}\right\} \triangleq \left\{C_1^D \cup \ldots \cup C_{2M}^D\right\} = \left\{C_1^{D-1} \cup \ldots \cup C_{4M}^{D-1}\right\} = \ldots = \left\{C_1^0 \cup \ldots \cup C_N^0\right\}$$

with centroids

$$\left\{(\lambda_1^j, \underline{\tilde{\boldsymbol{y}}}_1^j), \ldots, (\lambda_{2^{S-j}}^j, \underline{\tilde{\boldsymbol{y}}}_{2^{S-j}}^j)\right\} \quad \text{for} \quad 0 \leq j \leq D$$

whose centre of mass satisfies

$$\sum_{j=1}^{2^{S-j}} \lambda_k^j \underline{\tilde{\boldsymbol{y}}}_k^j = \underline{\tilde{\boldsymbol{b}}} \tag{1.53}$$

we can construct a sequence feasibility LP problems of the same form as (1.50).

In particular, starting from the coarsest level $j = D$, we have a collection of clusters $\{C_1, \ldots, C_{2M}\}$ with centroids

$$\{(\lambda_1^D, \underline{\tilde{\boldsymbol{y}}}_1^D), \ldots, (\lambda_{2M}^D, \underline{\tilde{\boldsymbol{y}}}_{2M}^D)\} \quad \text{for} \quad 0 \leq j \leq D$$

whose centre of mass satisfies

$$\sum_{j=1}^{2M} \lambda_k^D \underline{\tilde{\boldsymbol{y}}}_k^D = \underline{\tilde{\boldsymbol{b}}} \tag{1.54}$$

Therefore, we can formulate the following feasibility LP problem

$$C^{(D)} \underline{\boldsymbol{x}} = \underline{\tilde{\boldsymbol{b}}}, \quad C^{(D)} \triangleq [\underline{\tilde{\boldsymbol{y}}}_1^D, \ldots, \underline{\tilde{\boldsymbol{y}}}_{2M}^D] \in \mathbb{R}^{(M+1)\times 2M} \tag{1.55}$$

$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}$$

By construction, (1.55) admits a strictly feasible solution $\underline{\boldsymbol{\lambda}}^{(D)} = [\lambda_1^D, \ldots, \lambda_{2M}^D]^T$. Since by (1.54), we have

$$C^{(D)} \underline{\boldsymbol{\lambda}}^{(D)} = \underline{\tilde{\boldsymbol{b}}}, \quad \underline{\boldsymbol{\lambda}}^{(D)} > \underline{\mathbf{0}}$$

We apply the SVD-based algorithm, described in the previous section, to compute a basic feasible solution $\underline{\widehat{\boldsymbol{\lambda}}}^{(D)} = [\widehat{\lambda}_1^D, \ldots, \widehat{\lambda}_M^D, 0, \ldots, 0]^T$ to the LP problem in (1.55).

Once computed the basic feasible solution $\underline{\widehat{\boldsymbol{\lambda}}}^{(D)}$, we proceed to select the columns of C^D: $\{\underline{\tilde{\boldsymbol{y}}}_{i_1}^D, \ldots, \underline{\tilde{\boldsymbol{y}}}_{i_M}^D\} \subset \{\underline{\tilde{\boldsymbol{y}}}_1^D, \ldots, \underline{\tilde{\boldsymbol{y}}}_{2M}^D\}$, corresponding to the strictly positive weights $\{\widehat{\lambda}_1^D, \ldots, \widehat{\lambda}_M^D\}$ in $\underline{\widehat{\boldsymbol{\lambda}}}^{(D)}$. We also select the corresponding collection of subclusters $\{C_{i_1}^D, \ldots, C_{i_M}^D\} \subset \{C_1^D, \ldots, C_{2M}^D\}$.

Next, we re-weight this collection of subclusters

$$\{C_{i_1}^D, \ldots, C_{i_M}^D\} \to \{\widehat{C}_{i_1}^D, \ldots, \widehat{C}_{i_M}^D\}$$

by re-assigning the mass of the centroids of $C_{i_k}^D$ for $1 \leq k \leq M$

$$\lambda_k^D \to \widehat{\lambda}_k^D$$

whilst leaving the corresponding nodes of the centroids: $\underline{\tilde{\boldsymbol{y}}}_{i_k}^D$, unaltered.

Thereby, we produce an updated collection of clusters $\{\widehat{C}_{i_1}^D, \ldots, \widehat{C}_{i_M}^D\}$ with centroids $\{(\widehat{\lambda}_1^D, \underline{\tilde{\boldsymbol{y}}}_1^D), \ldots, (\widehat{\lambda}_M^D, \underline{\tilde{\boldsymbol{y}}}_{i_M}^D)\}$, whose centre of mass, satisfies

$$C^{(D)} \underline{\boldsymbol{\lambda}}^{(D)} = \sum_{k=1}^{M} \widehat{\lambda}_k^D \underline{\tilde{\boldsymbol{y}}}_{i_k}^D = \underline{\tilde{\boldsymbol{b}}}$$

by construction of the basic feasible solution $\widehat{\underline{\boldsymbol{\lambda}}}^{(D)} = [\widehat{\lambda}_1^D, \ldots, \widehat{\lambda}_M^D, 0, \ldots, 0]^T$ to the LP problem in (1.55).

Next, we endeavour to produce an updated hierarchical cluster structure with the coarsest level given by $\{\widehat{C}_{i_1}^D, \ldots, \widehat{C}_{i_M}^D\}$ and each cluster $\widehat{C}_{i_k}^D$, hierarchically subdivided into a collection of subclusters

$$\widehat{C}_{i_k}^D = \widehat{C}_{2i_k-1}^{D-1} \cup \widehat{C}_{2i_k}^{D-1} = \widehat{C}_{4i_k-3}^{D-2} \cup \widehat{C}_{4i_k-2}^{D-2} \cup \widehat{C}_{4i_k-1}^{D-2} \cup \widehat{C}_{4i_k}^{D-2} = \ldots$$

$$= \widehat{C}_{(i_k-1)2^D+1}^0 \cup \ldots \cup \widehat{C}_{i_k 2^D}^0$$

Moreover, we would like the centroids $(\widehat{\lambda}_\ell^j, \underline{\tilde{\boldsymbol{y}}}_{i_\ell}^j)$ of $\widehat{C}_\ell^j$ to satisfy a relationship

$$\sum_{\ell=1}^{M\cdot 2^{D-j}} \widehat{\lambda}_\ell^j \underline{\tilde{\boldsymbol{y}}}_{i_\ell}^j = \underline{\tilde{\boldsymbol{b}}}$$

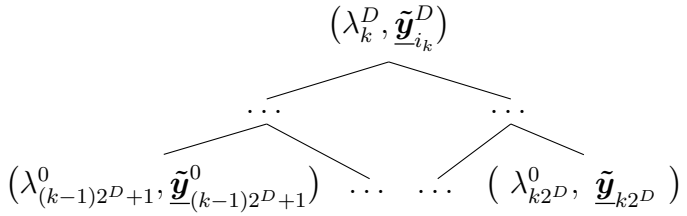
similar to (1.53).

To accomplish this, we proceed as follows. Take each cluster $C_{i_k}^D \in \{C_{i_1}^D, \ldots, C_{i_M}^D\}$ together with its constituent collection of subclusters

$$C_{i_k}^D = C_{2i_k-1}^{D-1} \cup C_{2i_k}^{D-1} = C_{4i_k-3}^{D-2} \cup C_{4i_k-2}^{D-2} \cup C_{4i_k-1}^{D-2} \cup C_{4i_k}^{D-2} = \ldots$$

$$= C_{(i_k-1)2^D+1}^0 \cup \ldots \cup C_{i_k 2^D}^0$$

Consider the binary tree representation of $C_{i_k}^D$, with tree nodes labelled by the centroids of $C_\ell^j \subset C_{i_k}^D$



We note that we can express each weight λ_ℓ^j on each level $0 \le j \le D-1$ of the tree, as a proportion of the root weight λ_k^D. Thus, for each level $0 \le j \le D-1$, we can write

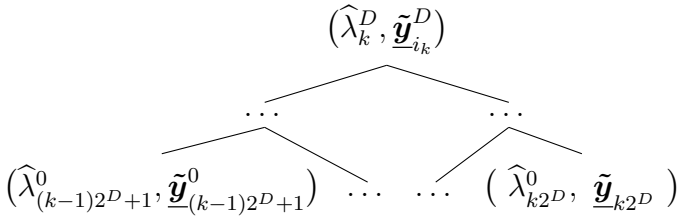
$$\lambda_\ell^j = \theta_\ell^j \lambda_k^D, \quad \text{where } \theta_\ell^j \triangleq \frac{\lambda_\ell^j}{\lambda_k^D} \quad \text{for } 1 \le \ell \le 2^{S-j}$$

We proceed to update the root weight on level $j = D$ and each weight λ_ℓ^j on every subsequent level $0 \leq j \leq D-1$

$$\lambda_k^D \to \widehat{\lambda}_k^D$$

$$\lambda_\ell^j = \theta_\ell^j \lambda_k^D \to \widehat{\lambda}_\ell^j = \theta_\ell^j \widehat{\lambda}_k^D$$

Thus, yielding a binary tree representation of $\widehat{C}_{i_k}^D$, with tree nodes labelled by the centroids of $\widehat{C}_\ell^j \subset \widehat{C}_{i_k}^D$



It can be readily verified that the updated tree represents a hierarchical clustering of $\widehat{C}_{i_k}^D$

$$\widehat{C}_{i_k}^D = \widehat{C}_{2i_k-1}^{D-1} \cup \widehat{C}_{2i_k}^{D-1} = \widehat{C}_{4i_k-3}^{D-2} \cup \widehat{C}_{4i_k-2}^{D-2} \cup \widehat{C}_{4i_k-1}^{D-2} \cup \widehat{C}_{4i_k}^{D-2} = \ldots$$

$$= \widehat{C}_{(i_k-1)2^D+1}^0 \cup \ldots \cup \widehat{C}_{i_k 2^D}^0$$

with each subcluster $\widehat{C}_\ell^j \subset \widehat{C}_{i_k}^D$ having centroids $(\widehat{\lambda}_\ell^j, \underline{\tilde{\boldsymbol{y}}}_\ell^j)$ satisfying

$$(\widehat{\lambda}_\ell^j, \underline{\tilde{\boldsymbol{y}}}_\ell^j) = \left(\widehat{\lambda}_{2\ell-1}^{j-1} + \widehat{\lambda}_{2\ell}^{j-1}, \frac{\widehat{\lambda}_{2\ell-1}^{j-1} \underline{\tilde{\boldsymbol{y}}}_{2\ell-1}^{j-1} + \widehat{\lambda}_{2k}^{j-1} \underline{\tilde{\boldsymbol{y}}}_{2\ell}^{j-1}}{\widehat{\lambda}_{2\ell-1}^{j-1} + \widehat{\lambda}_{2\ell}^{j-1}} \right)$$

and the centre of mass of the centroids satisfying

$$\sum_{\ell=1}^{M \cdot 2^{D-j}} \widehat{\lambda}_\ell^j \underline{\tilde{\boldsymbol{y}}}_{i_\ell}^j = \underline{\tilde{\boldsymbol{b}}}$$

Hence, we obtain the desired updated hierarchical clustering structure

$$\left\{ \widehat{C}_1^D \cup \ldots \cup \widehat{C}_{i_M}^D \right\} = \left\{ \widehat{C}_{2i_1-1}^{D-1} \cup \ldots \cup \widehat{C}_{2i_M}^{D-1} \right\} = \ldots = \left\{ \widehat{C}_{i_1}^0 \cup \ldots \cup \widehat{C}_{i_M 2^D}^0 \right\} \tag{1.56}$$

We note that, splitting each cluster in the coarsest hierarchical level $\left\{ \widehat{C}_1^D \cup \ldots \cup \widehat{C}_{i_M}^D \right\}$ into its two constituent subclusters of level $j = D-1$

$$\widehat{C}_{i_k}^D = \widehat{C}_{2i_k-1}^{D-1} \cup \widehat{C}_{2i_k}^{D-1}$$

does not alter the centre of mass of the centroids, since

$$\sum_{\ell=1}^{M} \widehat{\lambda}_{\ell}^{D} \underline{\tilde{\boldsymbol{y}}}_{i_\ell}^{D} = \sum_{\ell=1}^{M} \widehat{\lambda}_{2\ell-1}^{D-1} \underline{\tilde{\boldsymbol{y}}}_{2i_\ell-1}^{D-1} + \sum_{\ell=1}^{M} \widehat{\lambda}_{2\ell}^{D-1} \underline{\tilde{\boldsymbol{y}}}_{2i_\ell}^{D-1} = \underline{\tilde{\boldsymbol{b}}} \tag{1.57}$$

We proceed by splitting each cluster $\widehat{C}_{i_k}^{D}$ into its two constituent subclusters and setting

$$\left\{C_1, \ldots, C_{2M}\right\} \triangleq \left\{\widehat{C}_{2i_1-1}^{D-1}, \widehat{C}_{2i_1}^{D-1}, \ldots, \widehat{C}_{2i_M-1}^{D-1}, \widehat{C}_{2i_M}^{D-1}\right\}$$

The collection $\left\{C_1, \ldots, C_{2M}\right\}$ naturally inherits the hierarchical clustering structure of (1.56) for levels $0 \leq j \leq D-1$, with subclusters on each level j, having centroids

$$\left\{(\lambda_1^j, \underline{\tilde{\boldsymbol{y}}}_1^j), \ldots, (\lambda_{M\cdot 2^{D-j}}^j, \underline{\tilde{\boldsymbol{y}}}_{M\cdot 2^{D-j}}^j)\right\} \triangleq \left\{(\widehat{\lambda}_1^j, \underline{\tilde{\boldsymbol{y}}}_1^j), \ldots, (\widehat{\lambda}_{M\cdot 2^{D-j}}^j, \underline{\tilde{\boldsymbol{y}}}_{M\cdot 2^{D-j}}^j)\right\} \quad 0 \leq j \leq D-1$$

In particular, for the coarsest level $j = D-1$, we have a collection of clusters $\left\{C_1, \ldots, C_{2M}\right\}$ with centroids

$$\left\{(\lambda_1^{D-1}, \underline{\tilde{\boldsymbol{y}}}_1^{D-1}), \ldots, (\lambda_{2M}^{D-1}, \underline{\tilde{\boldsymbol{y}}}_{2M}^{D-1})\right\} \quad \text{for} \quad 0 \leq j \leq D-1$$

whose centre of mass satisfies

$$\sum_{j=1}^{2M} \lambda_k^{D-1} \underline{\tilde{\boldsymbol{y}}}_k^{D-1} = \underline{\tilde{\boldsymbol{b}}}$$

Therefore, analogously to (1.55), we formulate a feasibility LP problem

$$\begin{aligned} & C^{(D-1)} \underline{\boldsymbol{x}} = \underline{\tilde{\boldsymbol{b}}}, \quad C^{(D-1)} \triangleq [\underline{\tilde{\boldsymbol{y}}}_1^{D-1}, \ldots, \underline{\tilde{\boldsymbol{y}}}_{2M}^{D-1}] \in \mathbb{R}^{(M+1)\times 2M} \\ & \underline{\boldsymbol{x}} \geq \underline{\mathbf{0}} \end{aligned} \tag{1.58}$$

which by construction admits a strictly feasible solution $\underline{\boldsymbol{\lambda}}^{(D-1)} = [\lambda_1^{D-1}, \ldots, \lambda_{2M}^{D-1}]^T$.

We have completed the first iteration of the Recombination Algorithm, introduced in [96], for level $j = D$ and we proceed iterating the aforementioned procedure for levels $j = D-1, \ldots, 0$ and terminate at level $j = 0$. At each iteration, we construct an $(M+1) \times 2M$ feasibility LP problem with a strictly feasible solution and we construct a basic feasible solution via the SVD-Algorithm. We then, proceed to select the collection of M subclusters, corresponding to the strictly positive components of the basic feasible solution and re-weight this collection, so as to form an updated hierarchical cluster structure. Next, we proceed to split the resulting hierarchy of M clusters on

the coarsest level into $2M$ clusters and go back to constructing a feasibility LP problem.

By construction, solving the last feasibility LP problem for level $j = 0$

$$C^{(0)}\underline{\boldsymbol{x}} = \underline{\tilde{\boldsymbol{b}}}, \quad C^{(0)} \triangleq [\underline{\tilde{\boldsymbol{y}}}_1^0, \ldots, \underline{\tilde{\boldsymbol{y}}}_{2M}^0] = [\underline{\tilde{\boldsymbol{a}}}_{j_1}, \ldots, \underline{\tilde{\boldsymbol{a}}}_{j_{2M}}] \in \mathbb{R}^{(M+1)\times 2M} \tag{1.59}$$

$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}$$

yields a basic feasible solution $\underline{\widehat{\boldsymbol{\lambda}}}^{(0)} = [\widehat{\lambda}_1^0, \ldots, \widehat{\lambda}_M^0, 0, \ldots, 0]^T \in \mathbb{R}_+^{2M}$.

By selecting the columns of $C^{(0)}$: $\left\{\underline{\tilde{\boldsymbol{y}}}_{i_1}^0, \ldots, \underline{\tilde{\boldsymbol{y}}}_{i_M}^0\right\} = \left\{\underline{\tilde{\boldsymbol{a}}}_{j_{i_1}}, \ldots, \underline{\tilde{\boldsymbol{a}}}_{j_{i_M}}\right\}$, corresponding to the strictly positive weights $\{\widehat{\lambda}_1^0, \ldots, \widehat{\lambda}_M^0\}$ and setting $\mathcal{I}_B \triangleq \{j_{i_1}, \ldots, j_{i_M}\} \subset \{1, \ldots, N\}$ and $\mathcal{I}_N \triangleq \{1, \ldots, N\}/\mathcal{I}_B$, we can partition the constraint matrix $\tilde{A} = [\underline{\tilde{\boldsymbol{a}}}_1, \ldots, \underline{\tilde{\boldsymbol{a}}}_N] \in \mathbb{R}^{(M+1)\times N}$ in (1.47) as

$$\tilde{A} = [\tilde{B}, \tilde{N}] \quad \tilde{B} = [\underline{\tilde{\boldsymbol{a}}}_{j_{i_1}}, \ldots, \underline{\tilde{\boldsymbol{a}}}_{j_{i_M}}] \in \mathbb{R}^{(M+1)\times M}, \ \tilde{N} \in \mathbb{R}^{(M+1)\times(N-M)}$$

where $\tilde{B}$ consists of the subset of columns of $\tilde{A}$ indexed by $\mathcal{I}_B$, forming a linearly independent set, and $\tilde{N}$ consists of the subset of columns of $\tilde{A}$ indexed by $\mathcal{I}_N$.

Thereby, we can construct a basic feasible solution to (1.47) as

$$\underline{\widehat{\boldsymbol{\omega}}} = [\underline{\widehat{\boldsymbol{\omega}}}_B, \underline{\widehat{\boldsymbol{\omega}}}_N]^T = [\widehat{\omega}_1, \ldots, \widehat{\omega}_M, 0, \ldots, \ldots, 0] \triangleq [\widehat{\lambda}_1^0, \ldots, \widehat{\lambda}_M^0, 0, \ldots, \ldots, 0] \in \mathbb{R}_+^N \tag{1.60}$$

which satisfies

$$\tilde{A}\underline{\widehat{\boldsymbol{\omega}}} = \tilde{B}\underline{\widehat{\boldsymbol{\omega}}}_B + \tilde{N}\underline{\widehat{\boldsymbol{\omega}}}_N = \underline{\tilde{\boldsymbol{b}}} + \underline{\mathbf{0}} = \underline{\tilde{\boldsymbol{b}}}, \quad \underline{\widehat{\boldsymbol{\omega}}}_B > \underline{\mathbf{0}}, \ \underline{\widehat{\boldsymbol{\omega}}}_N = \underline{\mathbf{0}}$$

Analogously, we can partition the constraint matrix $A = [\underline{\boldsymbol{a}}_1, \ldots, \underline{\boldsymbol{a}}_N] \in \mathbb{R}^{M\times N}$ in (1.46) as

$$A = [B, N] \quad B = [\underline{\boldsymbol{a}}_{j_{i_1}}, \ldots, \underline{\boldsymbol{a}}_{j_{i_M}}] \in \mathbb{R}^{M\times M} \quad \text{non-singular}, \ \tilde{N} \in \mathbb{R}^{M\times(N-M)}$$

where B consists of the subset of columns of A indexed by $\mathcal{I}_B$ and N consists of the subset of columns of A indexed by $\mathcal{I}_N$. Thus, the basic feasible solution (1.60) for (1.46), satisfies

$$A\underline{\widehat{\boldsymbol{\omega}}} = B\underline{\widehat{\boldsymbol{\omega}}}_B + N\underline{\widehat{\boldsymbol{\omega}}}_N = \underline{\boldsymbol{b}} + \underline{\mathbf{0}} = \underline{\boldsymbol{b}}, \quad \underline{\widehat{\boldsymbol{\omega}}}_B > \underline{\mathbf{0}}, \ \underline{\widehat{\boldsymbol{\omega}}}_N = \underline{\mathbf{0}}$$

with $\underline{\widehat{\boldsymbol{\omega}}} = [\underline{\widehat{\boldsymbol{\omega}}}_B, \underline{\widehat{\boldsymbol{\omega}}}_N]^T$ with $\underline{\widehat{\boldsymbol{\omega}}}_B = B^{-1}\underline{\boldsymbol{b}}$, $\underline{\widehat{\boldsymbol{\omega}}}_N = \underline{\mathbf{0}}$. Hence, (1.60) is also a basic feasible solution for (1.46).

Algorithm 1: Recombination Algorithm

Input: $\{C_1, \ldots, C_{2M}\} \triangleq \{C_1^D, \ldots, C_{2M}^D\}$ with centroids $\{(\lambda_1^D, \underline{\tilde{\boldsymbol{y}}}_1^D), \ldots, (\lambda_{2M}^D, \underline{\tilde{\boldsymbol{y}}}_{2M}^D)\}$ and centre of mass of centroids $\underline{\tilde{\boldsymbol{b}}} = [\tilde{b}_1, \ldots, \tilde{b}_{M+1}]$

For $j = D : 0$

1) Construct a basic feasible solution $\underline{\widehat{\boldsymbol{\lambda}}}^{(j)} = [\widehat{\lambda}_1^j, \ldots, \widehat{\lambda}_M^j, 0, \ldots, 0]^T$ to

$$C^{(j)}\underline{\boldsymbol{x}} = \underline{\tilde{\boldsymbol{b}}}, \quad C^{(j)} \triangleq [\underline{\tilde{\boldsymbol{y}}}_1^j, \ldots, \underline{\tilde{\boldsymbol{y}}}_{2M}^j]$$

$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}$$

with a strictly feasible solution $\underline{\boldsymbol{\lambda}}^{(j)} = [\lambda_1^j, \ldots, \lambda_{2M}^j]^T$

2) Select subclusters $\{C_{i_1}^j, \ldots, C_{i_M}^j\} \subset \{C_1^j, \ldots, C_{2M}^j\}$ corresponding to strictly positive coordinates $\{\widehat{\lambda}_1^j, \ldots, \widehat{\lambda}_M^j\}$ of the basic feasible solution $\underline{\boldsymbol{\lambda}}^{(j)}$

3) Re-weight clusters $\{C_{i_1}^j, \ldots, C_{i_M}^j\} \to \{\widehat{C}_{i_1}^j, \ldots, \widehat{C}_{i_M}^j\}$ and respective subcusters, by re-assigning the mass of the corresponding centroids

$$\{(\lambda_1^r, \underline{\tilde{\boldsymbol{y}}}_1^r), \ldots, (\lambda_{i_M}^r, \underline{\tilde{\boldsymbol{y}}}_{i_{M\cdot 2^{D-j}}}^r)\} \to \{(\widehat{\lambda}_1^r, \underline{\tilde{\boldsymbol{y}}}_1^r), \ldots, (\widehat{\lambda}_{i_M}^r, \underline{\tilde{\boldsymbol{y}}}_{i_{M\cdot 2^{D-j}}}^r)\} \quad \text{for } 0 \leq r \leq j$$

4) Split each cluster in $\{\widehat{C}_{i_1}^j, \ldots, \widehat{C}_{i_M}^j\}$ into two constituent subclusters $\widehat{C}_{i_k}^j = \widehat{C}_{2i_k-1}^{j-1} \cup \widehat{C}_{2i_k}^{j-1}$ and set

$$\{C_1, \ldots, C_{2M}\} \triangleq \{\widehat{C}_{2i_1-1}^{j-1}, \widehat{C}_{2i_1}^{j-1}, \ldots, \widehat{C}_{2i_M-1}^{j-1}, \widehat{C}_{2i_M}^{j-1}\}$$

with centroids

$$\{(\widehat{\lambda}_{2i_1-1}^{j-1}, \underline{\tilde{\boldsymbol{y}}}_{2i_1-1}^{j-1}), \ldots, (\widehat{\lambda}_{2i_M}^{j-1}, \underline{\tilde{\boldsymbol{y}}}_{2i_M}^{j-1})\}$$

go to step 1)

Output: $\underline{\boldsymbol{\lambda}}^{(0)} = [\widehat{\lambda}_1^0, \ldots, \widehat{\lambda}_M^0, 0, \ldots, 0]^T$

We can apply the aforementioned algorithm to the feasibility LP problem

$$A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}} \tag{1.61}$$

$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}$$

where $A \in \mathbb{R}^{M\times N}, \boldsymbol{b} \in \mathbb{R}^N$ are defined in (1.36) and the vector of product cubature weights $\underline{\boldsymbol{\omega}} = [\omega_1, \ldots, \omega_N]^T \in \mathbb{R}^N_+$, arising from (1.35) is a strict feasible solution to (1.61), since

$$A\underline{\boldsymbol{\omega}} = \underline{\boldsymbol{b}}, \quad \underline{\boldsymbol{\omega}} > \underline{\mathbf{0}} \tag{1.62}$$

We note that, for an arbitrary matrix $A = [\underline{\boldsymbol{a}}_1, \ldots, \underline{\boldsymbol{a}}_N] \in \mathbb{R}^{M\times N}$, we would need to construct a transformed system of (1.61), to obtain $a_{1,j} = 1$ for $j = 1, \ldots, N$ and re-scale the strictly feasible solution $\underline{\boldsymbol{\omega}}$ to have mass one, as we did in transforming the system (1.46) to (1.47).

However, in the case of (1.61), we know that $A = [\gamma^d_m(\underline{\boldsymbol{x}}_1), \cdots, \gamma^d_m(\underline{\boldsymbol{x}}_N)]$, where $\gamma^d_m := (\gamma_1, \ldots, \gamma_M) : [0,1]^d \to \mathbb{R}^M$, which by definition satisfies, $\gamma_1(\underline{\boldsymbol{x}}) = 1$, $\forall \underline{\boldsymbol{x}} \in [0,1]^d$. Therefore, $a_{1,j} = \gamma(\underline{\boldsymbol{x}}_j) = 1$ for $j = 1, \ldots, N$ and furthermore the first equation of (1.61) yields $\sum_{i=1}^N \omega_i = 1$. Therefore, no transformation is required and thus (1.61) is of the same form as (1.47), with the trivial objective function. Hence, we proceed, as previously described, with the hierarchical cluster representation of (1.62) and an application of the Recombination Algorithm to (1.61), to obtain a basic feasible solution to (1.61).

1.3.3.5 Computational Complexity of the Carathéodory Cubature with SVD problem formulation

As noted earlier, the computational complexity of constructing $Q^{CAR}(m,d)$ is dominated by the construction of the Cartesian product formula in (1.35), constructing the linear system (1.61) and finding the basic feasible solution to (1.61), via hierarchical clustering and an application of the Recombination Algorithm. Since, the Recombination Algorithm utilises the SVD-based algorithm for computing basic feasible solutions to a sequence of smaller LP problems, we denote the overall cost of finding the basic feasible solution to (1.61) by $C_{SVD}(M,N)$.

Hence, the complexity of computing $Q^{CAR}(m,d)$ with SVD problem formulation is given by

$$C\Big(Q^{CAR}(m,d)\Big) = O\Big(NM + C_{SVD}(M,N)\Big)$$

where

$$C_{SVD}(M,N) = O\Big(NM + \log_2\Big(\frac{N}{2M}\Big)M^3\Big)$$

The cost associated with hierarchical clustering of the data in (1.62) is of $O(NM)$. We compute this cost, as follows, starting from level $j=0$ containing N pairs $\Big\{(\tilde{\omega}_i, \underline{\tilde{\boldsymbol{a}}}_i)\Big\}_{i=1}^{N}$ and forming each successive level $1 \leq j \leq D$, $D := \log_2\big(\frac{N}{2M}\big)$ by $\frac{N}{2^j}$ multiplications by a scalar and additions of two vectors of length M. Thus, adding over all D levels of the construction yields

$$2M\sum_{j=1}^{D}\frac{N}{2^j} \leq \frac{7}{4}NM$$

We recall that the Recombination Algorithm requires $D+1$ iterations through levels $j=0,\ldots,D$, with the cost of each iteration, dominated by the computation of the basic feasible solution to the a feasibility LP problem. The latter operation is performed via the SVD-based Algorithm, which has an associated complexity of $O(M^3)$. Therefore, $(D+1)$ iterations of computations of order $O(M^3)$, yield the second the term in the complexity of $C_{SVD}(M,N)$.

Thus, the overall cost of constructing $Q^{CAR}(m,d)$ with SVD problem formulation is given by

$$C\Big(Q^{CAR}(m,d)\Big) = O\Big(NM + \log_2\Big(\frac{N}{2M}\Big)M^3\Big)$$

The cost $O(NM)$ associated with the initial Cartesian product cubature construction and hierarchical clustering can reduced via a recursive algorithm to $O(M^2)$, independent of N, as will be illustrated in Chapter 4. Thus, without explicit proof, we conclude

that the theoretical cost of constructing $Q^{CAR}(m,d)$ with SVD problem formulation is given by

$$C\Big(Q^{CAR}(m,d)\Big) = O\Big(M^3\Big)$$

we note that dependence on N, the number of nodes in the Cartesian product cubature $Q(m,d)$, that grows exponentially with dimension, is only present implicitly in the $\log_2\left(\frac{N}{2M}\right)$ constant multiplier of M^3. Moreover, there is no quadratic dependence on N as in the cost of constructing $Q^{CAR}(m,d)$ with Simplex problem formulation

$$C\Big(Q^{CAR}(m,d)\Big) = O(M^3N + MN^2)$$

We recall that, since in the vast majority of cases $N >> M$, hence this reduction of dependence of complexity on N is crucial to improve the runtime of the overall algorithm. As the complexity of the Carathéodory Cubature construction with SVD problem formulation is only dependent on $M = \binom{m+d}{d}$, we conclude that the cost grows polynomially, rather than exponentially with dimension d.

Moreover, the novel SVD-based algorithm for computing basic feasible solutions with a prescribed strictly feasible solutions, presented in Section 1.3.3.3, produces an order of magnitude speed-up $O(M^4) \rightarrow O(M^3)$ of the overall Carathéodory Cubature construction algorithm, when compared to the algorithm employed in [96].

1.3.4 Carathéodory cubature precision

Similarly to the Smolyak construction, the Carathéodory cubature construction will depend on the choice of constituent one-dimensional quadrature formulae in (1.33). However, unlike the Sparse Grid methods, the Carathéodory Cubature Algorithm does not benefit from nestedness of the underlying sequence of quadrature formulae. Moreover, unlike the Smolyak Algorithm, the Carathéodory Cubature Algorithm will always produce a cubature formula with strictly positive weights and of norm $\|Q^{CAR}(m,d)\| = \sum_{i=1}^{M'} \lambda_i = 1$, provided that the underlying sequence of univariate quadrature formulae has positive weights and are of norm one. Whence, theoretically the stability of Carathéodory Cubature Algorithm, should not be affected by the choice of the quadrature sequence.

Similarly to the Cartesian product cubature $Q(m,d)$ of univariate quadratures $\{Q_i\}_{i=1}^{d}$ of $deg(Q_i) = m$, we have

$$deg\big(Q^{CAR}(m,d)\big) = deg\big(Q(m,d)\big) = m$$

Thus, given an integrand f in (1.32), satisfying

$$\Big|\frac{\partial^m f}{(\partial x^i)^m}\Big| \leq M \qquad \text{for} \quad i \in [\![1,d]\!] \tag{1.63}$$

for some constant M and $m \in \mathbb{N}$, so that, in each variable f is $C^m([0,1]^d)$ uniformly with respect to the other variables, the burden of utilising all the underlying smoothness of f falls on the cubature formula, which should ideally have the degree of precision $m \in \mathbb{N}$, with the fewest abscissae.

Therefore, by construction of Carathéodory Cubature formula $Q^{CAR}(m,d)$, the best results in terms of precision and cardinality of the resulting cubature formula are obtained when Gauss-Legendre quadrature formulae, described in Section 1.1.3.1, are utilised as the constituent sequence of univariate quadratures forming the input measure into the Recombination Algorithm $Q(m,d)$.

By condition (1.63), there exists a constant K such that

$$\Big|\int_{[0,1]} f(x^1,\ldots,x^d)dx^i - Q[f,x^i]\Big| \leq K\Big(\lfloor\frac{m}{2}\rfloor + 1\Big)^{-m} \qquad \text{for} \quad i \in [\![1,d]\!]$$

for all values of $x^1, x^2, \ldots, x^{i-1}, x^{i+1}, \ldots, x^d$ lying between 0 and 1, where $Q[f,x^i]$ for $1 \leq i \leq d$ are identical univariate Gauss-Legendre formulae of degree $m \in \mathbb{N}$ over

$[0,1]$. The above error estimate is the same as the one provided in (1.18) for cubature product measure and the underlying univariate Gauss-Legendre formulae are given by

$$Q[f,x^i] = \sum_{i=1}^{\lfloor \frac{m}{2} \rfloor + 1} \lambda_j^i f(x_j^i) \qquad x_j^i := \frac{1}{2} x_j^* + \frac{1}{2}, \quad \lambda_j^i := \lambda_j^*$$

where x_j^* are the zeros of the $\Big(\lfloor \frac{m}{2} \rfloor + 1\Big)^{th}$ order Legendre polynomial $L_{\lfloor \frac{m}{2} \rfloor + 1}(x)$, given in Section 1.1.3.1 and $\lambda_j^* = 1/(1-x_j^2)[L'_{\lfloor \frac{m}{2} \rfloor + 1}(x_j^*)]^2$.

By setting $A := \sum_{j=1}^{\lfloor \frac{m}{2} \rfloor + 1} \lambda_j^* = 1$, similarly to the Cartesian product cubature $Q(m,d)$, we obtain the following error estimate for $Q^{CAR}(m,d)$

$$\begin{aligned} &\big|R^{CAR}(m,d)[f]\big| = \big|I_d[f] - Q^{CAR}(m,d)[f]\big| \\ &\leq (1 + A + \ldots + A^{d-1}) K \Big(\lfloor \frac{m}{2} \rfloor + 1 \Big)^{-m} = K d \Big(\lfloor \frac{m}{2} \rfloor + 1 \Big)^{-m} \end{aligned}$$

Moreover, as remarked earlier Bahvalov has demonstrated that this upper error bound cannot be improved upon for the class of functions satisfying condition (1.63).

Thus, for a fixed smoothness $m \in \mathbb{N}$ of the underlying integrand, given by condition (1.63), the best error estimate on Carathéodory Cubature formula $Q^{CAR}(m,d)$ is given by

$$O\Big(\lfloor \frac{m}{2} \rfloor + 1 \Big)^{-m}$$

We recall that whilst for a fixed smoothness $m \in \mathbb{N}$ of the underlying integrand f, the error for both $Q^{CAR}(m,d)$ and $Q(m,d)$ is of the same order of magnitude, the cardinalities are given by

$$\begin{aligned} card\Big(Q(m,d)\Big) &= \Big(\lfloor \frac{m}{2} \rfloor + 1 \Big)^d \\ card\Big(Q^{CAR}(m,d)\Big) &\leq \binom{m+d}{d} \end{aligned}$$

and therefore clearly, the number of function evaluations required increases respectively exponentially and polynomially with dimension d.

The following table provides some idea as to the magnitude of $card\Big(Q^{CAR}(m,d)\Big)$ and $card\Big(Q(m,d)\Big)$. We compare the number of cubature nodes in $Q^{CAR}(m,d)$ and $Q(m,d)$ for a fixed dimension $d = 5$ for degrees of precision $m = 1, \ldots, 16$.

DEG	$card\Big(Q(m,5)\Big)$	$\binom{m+d}{d}$	$card\Big(Q^{CAR}(m,5)\Big)$
1	1	6	1
2	32	21	21
3	32	56	21
4	243	126	126
5	243	256	126
6	1024	462	461
7	1024	792	461
8	3125	1287	1286
9	3125	2002	1286
10	7776	3003	3002
11	7776	4368	3002
12	16807	6168	6168
13	16807	8568	6168
14	32768	11628	11627
15	32768	15504	11627
16	59049	20349	20349

Table 1.2: Cardinality of full tensor Gaussian Cubature and Carathéodory Gaussian Cubature

1.4 Primal, Dual Simplex Algorithms and Interior Point Method

1.4.0.1 Primal Simplex Algorithm

In the forthcoming section, we will illustrate how to obtain a basic feasible solution to (1.39), via a variety of Linear Programming methods: The Primal Simplex Algorithm, The Dual Simplex Algorithm and the Primal-Dual Interior Point Method. We begin by introducing the Primal Simplex Algorithm; we recall that (1.39) is given by

$$A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}$$

$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}$$

where $A := [\underline{\boldsymbol{a}}_1, \ldots, \underline{\boldsymbol{a}}_N] = [\gamma_m^d(\underline{\boldsymbol{x}}_1), \cdots, \gamma_m^d(\underline{\boldsymbol{x}}_N)] \in \mathbb{R}^{M \times N}$, $\underline{\boldsymbol{b}} \in \mathbb{R}_+^M$ and we already have a strictly feasible solution $\underline{\boldsymbol{\omega}} := [\omega_1, \ldots, \omega_N]^T \in \mathbb{R}_+^N$, satisfying $A\underline{\boldsymbol{\omega}} = \underline{\boldsymbol{b}}$, $\underline{\boldsymbol{\omega}} \geq \underline{\mathbf{0}}$.

In section (1.3.1) we highlighted that since (1.39) admits a feasible solution $\underline{\boldsymbol{\omega}}$, it also admits a BFS (basic feasible solution) $\widehat{\underline{\boldsymbol{\omega}}}$. Moreover, we recall that each BFS corresponds to an extreme point of the feasible region and hence there exist potentially up to $\binom{N}{M}$ BFSs to (1.39). A commonly employed method to obtain one basic feasible solution, is to introduce some artificial constraints into (1.39), thus yielding a re-formulated system of equations and subsequently employ the Simplex Algorithm to force these artificial variables to zero, see for example Chapter 4 of [50].

Assume that $rank(A) = M$, to be relaxed later and suppose that we add a non-negative artificial vector $\underline{\boldsymbol{x}}_a \in \mathbb{R}_+^M$ to the constraints in (1.39), thus resulting in the following linear system

$$A\underline{\boldsymbol{x}} + \underline{\boldsymbol{x}}_a = \underline{\boldsymbol{b}}$$

$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}},\ \underline{\boldsymbol{x}}_a \geq \underline{\mathbf{0}}$$

which we re-formulate in the standard form by setting

$$D\underline{\tilde{\boldsymbol{x}}} = \underline{\boldsymbol{b}} \tag{1.64}$$

$$\underline{\tilde{\boldsymbol{x}}} \geq \underline{\mathbf{0}}$$

where $D = [A, I] \in \mathbb{R}^{M\times(N+M)}$, $\underline{\tilde{\boldsymbol{x}}} = [\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}_a] \in \mathbb{R}^{(N+M)}$

By construction we have an immediate BFS to the newly formulated LP problem (1.64) namely $\underline{\tilde{\boldsymbol{x}}} = [\underline{\mathbf{0}},\ \underline{\boldsymbol{b}}]$, by setting $\underline{\boldsymbol{x}} = \underline{\mathbf{0}}$ and $\underline{\boldsymbol{x}}_a = \underline{\boldsymbol{b}}$.

However, in order to get back to our original problem, we must force the artificial variables to zero, since $A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}$ if and only if $A\underline{\boldsymbol{x}} + \underline{\boldsymbol{x}}_a = D\underline{\tilde{\boldsymbol{x}}} = \underline{\boldsymbol{b}}$ with $\underline{\boldsymbol{x}}_a = \underline{\mathbf{0}}$. A widely employed method to accomplish this, is to minimise the sum of the artificial variables, which amounts to solving the following LP problem

$$\begin{aligned} &\text{Minimise} \quad \underline{\boldsymbol{c}}^T\underline{\tilde{\boldsymbol{x}}} \\ &\text{subject to} \quad D\underline{\tilde{\boldsymbol{x}}} = \underline{\boldsymbol{b}} \\ &\qquad\qquad \underline{\tilde{\boldsymbol{x}}} \geq \underline{\mathbf{0}} \end{aligned} \tag{1.65}$$

where $\underline{\boldsymbol{c}} = [0, \ldots, 0, 1, \ldots, 1]$ and thus $\underline{\boldsymbol{c}}^T\underline{\tilde{\boldsymbol{x}}} = \sum_{i=1}^{M}(\underline{\boldsymbol{x}}_a)_i$.

We can solve (1.65), starting with BFS $\underline{\tilde{\boldsymbol{x}}} = [\underline{\mathbf{0}},\ \underline{\boldsymbol{b}}]$ and employing the Simplex Algorithm, which will be described shortly. At optimality of (1.65), we will have $\underline{\boldsymbol{x}}_a = \underline{\mathbf{0}}$ and hence we have computed $\underline{\boldsymbol{x}}$ a BFS to (1.39).

We note that if the original problem (1.39) has a feasible solution, then the optimal value of (1.65) is zero. Thus, as we already have a feasible solution $\underline{\boldsymbol{\omega}}$ to the original problem; we are certain to achieve optimality in (1.65).

We are now in a position to apply the Simplex Algorithm to (1.65) and thus find a basic feasible solution to (1.39). The Simplex Algorithm we are about to describe, is the widely employed, two-phase Simplex Algorithm, see [49], [50], in which phase

(I) consists of finding an initial basic feasible solution and phase (II) consists of sequentially moving from the current basic feasible solution to an adjacent basic feasible solution with an improving direction, until we have found the optimal solution.

Since an initial BFS to (1.65) is given by $\underline{\tilde{\boldsymbol{x}}}^{(0)} = [\underline{\mathbf{0}},\ \underline{\boldsymbol{b}}]$, we readily proceed to phase (II) of the Simplex Algorithm. To proceed, we need introduce the notions of adjacent BFSs, feasible directions and improving directions of the feasible region

$$\mathcal{X} := \left\{ \underline{\boldsymbol{x}} \in \mathbb{R}^{(N+M)} \ : \ D\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}, \ \ \underline{\boldsymbol{x}} \geq \underline{\mathbf{0}} \right\}$$

associated with LP problem (1.65).

Geometrically, two extreme points of the feasible region $\mathcal{X}$ are called adjacent if the line segment joining them is an edge of $\mathcal{X}$. Or equivalently, in algebraic terms two basic feasible solutions $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$ and $\underline{\tilde{\boldsymbol{x}}}^{(\ell+1)}$ are called adjacent if their bases $\mathcal{B}^{(\ell)}$ and $\mathcal{B}^{(\ell+1)}$ differ by exactly one element.

Now, we introduce the notion of a feasible direction. A non-zero vector $\underline{\boldsymbol{d}}$ is a feasible direction at $\underline{\boldsymbol{x}} \in \mathcal{X}$, if there is some $\widehat{\lambda} > 0$ such that $\underline{\boldsymbol{x}} + \lambda \underline{\boldsymbol{d}}$ is feasible for all $0 < \lambda \leq \widehat{\lambda}$. Additionally, it is possible that along a feasible direction $\underline{\boldsymbol{d}}$, $\underline{\boldsymbol{x}} + \lambda \underline{\boldsymbol{d}}$ is feasible for all $\lambda > 0$.

Thus $\underline{\boldsymbol{d}}$, $\underline{\boldsymbol{d}} \neq \underline{\mathbf{0}}$ is a feasible direction at $\underline{\boldsymbol{x}} \in \mathcal{X}$ if and only if

$$D(\underline{\boldsymbol{x}} + \lambda \underline{\boldsymbol{d}}) = \underline{\boldsymbol{b}}$$

$$\underline{\boldsymbol{x}} + \lambda \underline{\boldsymbol{d}} \geq \underline{\mathbf{0}}$$

for each $0 < \lambda \leq \widehat{\lambda}$. We note that for $\underline{\boldsymbol{d}}$ to be a feasible direction, we must to have $D\underline{\boldsymbol{d}} = \underline{\mathbf{0}}$.

The fundamental question of phase (II) of the Simplex Algorithm, is, given a current BFS: $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$, how do we find a feasible direction $\underline{\boldsymbol{d}}$ that improves the value of our objective function $f(\underline{\tilde{\boldsymbol{x}}}) = \underline{\boldsymbol{c}}^T \underline{\tilde{\boldsymbol{x}}}$. To answer this, we introduce the notion of an improving direction $\underline{\boldsymbol{d}}$.

A vector $\underline{\boldsymbol{d}}$, $\underline{\boldsymbol{d}} \neq \underline{\mathbf{0}}$ is an improving direction at the current BFS: $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$, if there exists a $\widehat{\lambda} > 0$, such that the objective function value of

$$\underline{\tilde{\boldsymbol{x}}}^{(\ell+1)} := \underline{\tilde{\boldsymbol{x}}}^{(\ell)} + \lambda \underline{\boldsymbol{d}}$$

is better than that of $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$ for all $0 < \lambda \leq \widehat{\lambda}$.

Given that problem (1.65) is a minimisation problem, we have that the objective function value at $\underline{\tilde{\boldsymbol{x}}}^{(\ell+1)}$ is better than the objective function value at $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$, if

$$f(\underline{\tilde{\boldsymbol{x}}}^{(\ell+1)}) < f(\underline{\tilde{\boldsymbol{x}}}^{(\ell)})$$

Thus a vector $\underline{\boldsymbol{d}}$ is an improving direction at $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$, if it satisfies

$$\nabla f(\underline{\tilde{\boldsymbol{x}}}^{(\ell)})^T \underline{\boldsymbol{d}} = \underline{\boldsymbol{c}}^T \underline{\boldsymbol{d}} < 0$$

where $\nabla f(\underline{\tilde{\boldsymbol{x}}}^{(\ell)})$ denotes the gradient of f at $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$.

Let us assume that we are currently at a BFS to (1.65): $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$ with basis $\mathcal{B}^{(\ell)} = \{\tilde{x}_1^{(\ell)}, \ldots, \tilde{x}_M^{(\ell)}\}$. We know that $\underline{\tilde{\boldsymbol{x}}}^{(\ell)} = [\underline{\tilde{\boldsymbol{x}}}_B^{(\ell)}, \underline{\tilde{\boldsymbol{x}}}_N^{(\ell)}]^T$, where $\underline{\tilde{\boldsymbol{x}}}_B^{(\ell)} = [\tilde{x}_1^{(\ell)}, \ldots, \tilde{x}_M^{(\ell)}] \in \mathbb{R}_+^M$ is a vector of basic variables and $\underline{\tilde{\boldsymbol{x}}}_N^{(\ell)} = [\tilde{x}_{M+1}^{(\ell)}, \ldots, \tilde{x}_{M+N}^{(\ell)}] \in \mathbb{R}_+^N$, is a vector of non-basic variables, satisfying

$$\underline{\tilde{\boldsymbol{x}}}_B^{(\ell)} = B^{-1}\underline{\boldsymbol{b}}$$

$$\underline{\tilde{\boldsymbol{x}}}_N^{(\ell)} = \underline{\mathbf{0}}$$

where $B = [\underline{\boldsymbol{d}}_{j_1}, \ldots, \underline{\boldsymbol{d}}_{j_M}]$ with $rank(B) = M$, $\underline{\boldsymbol{d}}_{j_k}$ is a column of D.

And further assume that we wish to move to an adjacent BFS: $\underline{\tilde{\boldsymbol{x}}}^{(\ell+1)}$ with basis $\mathcal{B}^{(\ell+1)}$. Then, since $\mathcal{B}^{(\ell)}$ and $\mathcal{B}^{(\ell+1)}$ differ by exactly one element, there is one non-basic variable of $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$, that is in $\mathcal{B}^{(\ell+1)}$.

Assume that the non-basic variable $\tilde{x}_k^{(\ell)} \in \underline{\tilde{\boldsymbol{x}}}_N^{(\ell)}$ is to become basic, then the feasible direction $\underline{\boldsymbol{d}}$, must have $d_k > 0$, since we are increasing the value of $\tilde{x}_k^{(\ell)}$ from zero. Without loss of generality, we may assume that $d_k = 1$ and since all other non-basic variables $\tilde{x}_j^{(\ell)} \in \underline{\tilde{\boldsymbol{x}}}_N^{(\ell)}$, $j \neq k$ remain non-basic, we have $d_j = 0$. We compute the remaining components d_i, corresponding to the basic variables $\tilde{x}_i^{(\ell)} \in \underline{\tilde{\boldsymbol{x}}}_B^{(\ell)}$ by solving $D\underline{\boldsymbol{d}} = \underline{\mathbf{0}}$. Thus, we have the following definition.

Definition 1.4.1. *A feasible simplex direction* $\underline{\boldsymbol{d}}^k$ *corresponding to the non-basic variable* $\underline{\tilde{\boldsymbol{x}}}_k^{(\ell)}$ *of a BFS:* $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$*, is a vector, where*

$$1) \quad d_j^k = 0, \quad \textit{for} \quad j \in \mathcal{I}_{\mathcal{N}}, \quad j \neq k \quad \textit{and} \quad d_k^k = 1$$

$$2) \quad d_i^k, \quad \textit{for} \quad i \in \mathcal{I}_{\mathcal{B}}, \quad \textit{is determined by solving } D\underline{\boldsymbol{d}} = \underline{\mathbf{0}}$$

where $\mathcal{I}_{\mathcal{N}}$ *is the set of indices of non-basic variables of* $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$ *and* $\mathcal{I}_{\mathcal{B}}$ *is the set of indices of basic variables of* $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$*.*

By utilising definition (1.4.1), we can now compute all possible feasible directions $\underline{\boldsymbol{d}}^k$, $k \in \mathcal{I}_{\mathcal{N}}$ to move us to an adjacent BFS.

Subsequently, we will need to determine whether any of these directions are improving. To accomplish this for each feasible simplex direction $\underline{\boldsymbol{d}}^k$, we compute the associated reduced cost, as defined in the following.

Definition 1.4.2. *The reduced cost* $\bar{c}_k$ *associated with the non-basic variable* $\underline{\tilde{\boldsymbol{x}}}_k^{(\ell)}$ *of a BFS* $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$ *is given by*

$$\bar{c}_k = \underline{\boldsymbol{c}}^T \underline{\boldsymbol{d}}^k$$

where $\underline{\boldsymbol{d}}^k$ *is the simplex direction associated with variable* $\underline{\tilde{\boldsymbol{x}}}_k^{(\ell)}$*.*

Now, we know that the feasible simplex direction $\underline{\boldsymbol{d}}^k$ is an improving direction for a minimisation problem if $\bar{c}_k < 0$. Whilst, no negative reduced cost $\bar{c}_k$ has been proven to be a better choice than another negative reduced cost $\bar{c}_j$, theoretically, computational experiments suggest that the one with the largest (negative) reduced cost should be selected, this rule known as Dantzig rule. In the event that there are no improving simplex directions, we have obtained the optimal solution.

If the optimal solution has not yet been found, then we proceed to choose an improving direction $\underline{\boldsymbol{d}}$. We note that we cannot have $\underline{\boldsymbol{d}} \geq \underline{\mathbf{0}}$, since it would imply that $\underline{\tilde{\boldsymbol{x}}}^{(\ell)} + \lambda \underline{\boldsymbol{d}} \geq \underline{\mathbf{0}}$ is feasible for all $\lambda \geq 0$. As a consequence, the optimisation problem (1.65) would be unbounded, as noted in Chapter 4 [51]. Since, we know that the LP problem (1.65) achieves an optimal solution, it cannot be unbounded.

Thus, we proceed to determine the maximum step size or how far we move in our chosen improving simplex direction $\underline{\boldsymbol{d}}$. Since there exists $j \in \{1, \ldots, M+N\} : \ d_j < 0$, we can define

$$\lambda_{max} = \min \left\{ \frac{\tilde{x}_j^{(\ell)}}{-d_j} \ : \ d_j < 0 \right\}$$

Then, for all $i \in \{1, \ldots, M+N\}$, we have

$$\tilde{x}_i^{(\ell)} + \lambda d_i \geq 0 \qquad \text{for all} \quad 0 < \lambda \leq \lambda_{max} \tag{1.66}$$

If $d_i > 0$ (1.66) holds for all $\lambda > 0$. Whilst, if $d_i < 0$, then

$$\tilde{x}_i^{(\ell)} + \lambda d_i \geq \tilde{x}_i^{(\ell)} + \lambda_{max} d_i \geq 0$$

holds by definition of λ_{max}. Thus, we produce an updated BFS

$$\underline{\tilde{\boldsymbol{x}}}^{(\ell+1)} := \underline{\tilde{\boldsymbol{x}}}^{(\ell)} + \lambda_{max} \underline{\boldsymbol{d}}$$

and conclude one iteration through phase (II) of the Simplex Algorithm.

Subsequently, we return to construct all possible feasible simplex directions for $\underline{\tilde{\boldsymbol{x}}}^{(\ell+1)}$ and iterate the aforementioned process, until an optimal solution is found. To summarise we present the two-phase Simplex Algorithm.

Algorithm 2: Simplex Algorithm

Phase (I):

Step (0) Initialisation: Identify an initial BFS: $\underline{\tilde{\boldsymbol{x}}}^{(0)}$ and set $\ell = 0$

Phase (II):

Step (1) Construct Simplex Directions: $\underline{\boldsymbol{d}}^k$ and Reduced Costs: $\bar{c}_k$,

for each $k \in \mathcal{I}_{\mathcal{N}}$, where $\mathcal{I}_{\mathcal{N}}$ is the set of indices corresponding to non-basic variables of $\underline{\tilde{\boldsymbol{x}}}^{(\ell)}$.

Step (2) Optimality Check: If no Simplex Direction is improving, stop. The current solution $\underline{\tilde{\boldsymbol{\beta}}}^{(\ell)}$ is optimal. Else, choose an improving simplex direction $\underline{\boldsymbol{d}}$.

Step (3) If $\underline{\boldsymbol{d}} \geq \underline{\mathbf{0}}$, stop. LP problem is unbounded. Else, compute maximum step size:

$$\lambda_{max} = \min \left\{ \frac{\tilde{x}_j^{(\ell)}}{-d_j} \ : \ d_j < 0 \right\}$$

Step (4) Compute new BFS

$$\underline{\tilde{\boldsymbol{x}}}^{(t+1)} = \underline{\tilde{\boldsymbol{x}}}^{(t)} + \lambda_{max} \underline{\boldsymbol{d}}$$

set $\ell = \ell + 1$ and return to step (1)

We shall now provide some theoretical considerations as to the performance of the Simplex Algorithm. The performance of the Simplex Algorithm can be broadly divided into two types: worst case analysis and average case analysis. A worst case analysis looks at all problems of a given size and asks how much effort is needed to solve the hardest of these. Whilst, the average case analysis looks at the effort required averaging over all problems of a given size.

The Simplex Algorithm operates by finding a basic feasible solution or extreme point of the feasible region and then by successively moving to an improved basic feasible solution until the optimal point is reached or unboundedness of the objective function is verified. Hence an upper bound on the number of iterations is the number of

extreme points in the feasible region in (1.39), that is $\binom{N}{M}$. Klee-Minty provided an example of an LP problem in n variables with n constraints, whose feasible region is an n-dimensional hypercube with 2^n vertices. For this problem, the Simplex algorithm starts at one of these vertices and visits every vertex before finding the optimal solution, requiring $2^n - 1$ iterations, see [[49], Chapter 4]. In fact, it still an open question as to whether there are pivot rules for the Simplex algorithm, for which one can prove, that no problem instance exists which requires an exponential number of iterations, as a function M and L. In practice, however, the Simplex method takes on average O(L) iterations to find an optimal solution.

The cost of each step of the Simplex Method to move from one basic feasible solution to the next is generally assumed to be of order $O(M^3 + NM)$, dominated by the computation of $\underline{\tilde{\boldsymbol{x}}}_B = B^{-1}\underline{\boldsymbol{b}}$ of order $O(M^3)$. A version of the Simplex algorithm called the revised Simplex algorithm takes less time since its does not recompute the basis matrix B^{-1} from scratch every time, but it uses the fact that the basis matrices B differ by only one column from one iteration to the next and hence costs only $O(M^2 + NM)$. We note that there is no universal consensus in the LP literature as to the precise complexity of the Simplex algorithm, but rather worst and average case analysis estimates exist for a multitude of the various implementations. From the average runtime complexities just mentioned, we can derive an upper and lower bound on the cost of the Simplex method, depending on whether the ordinary or the revised Simplex Method is implemented

$$O(NM^2 + N^2M) \leq C_{SIMPL}\big(M, N\big) \leq O(NM^3 + N^2M)$$

Throughout this section we have implicitly assumed that $rank(A) = M$ in the LP problem (1.65). Now we consider the scenario in which $rank(A) = M^{'} < M$, this frequently occurs in the context of computations of Carathéodory cubature formulae.

Example 1.4.3. *Suppose that* $A \in \mathbb{R}^{3\times 4}$ *and* $\underline{\boldsymbol{b}} \in \mathbb{R}^3_+$ *in* (1.65) *are given by*

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \qquad \underline{\boldsymbol{b}} = \begin{bmatrix} 1 \\ 0.5 \\ 1 \end{bmatrix}$$

then clearly, $rank(A) = 2 < 3$ *and the defining* D *as in* (1.64)*, we obtain*

$$D = [\underline{\boldsymbol{d}}_1,\ \underline{\boldsymbol{d}}_2,\ \underline{\boldsymbol{d}}_3,\ \underline{\boldsymbol{d}}_4,\ \underline{\boldsymbol{d}}_5,\ \underline{\boldsymbol{d}}_6,\ \underline{\boldsymbol{d}}_7] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Let us consider the basic feasible solution with basis $\mathcal{B}_1 = \{\underline{\boldsymbol{d}}_1,\ \underline{\boldsymbol{d}}_2,\ \underline{\boldsymbol{d}}_5\}$

$$\underline{\tilde{\boldsymbol{x}}}^1_B = \begin{bmatrix} \tilde{x}^1_1 \\ \tilde{x}^1_2 \\ \tilde{x}^1_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 0.25 \\ 0 \end{bmatrix}, \quad \underline{\tilde{\boldsymbol{x}}}^1_N = \tilde{x}^1_4 = 0$$

We note that the basic feasible solution $\underline{\tilde{\boldsymbol{x}}^1} = [\ \underline{\tilde{\boldsymbol{x}}}^1_B, \underline{\tilde{\boldsymbol{x}}}^1_N\ ]^T$ *is degenerate since the variable* $\tilde{x}^1_4 = 0$*. Now, we consider the basic feasible solution associated with basis* $\mathcal{B}_2 = \{\underline{\boldsymbol{d}}_1,\ \underline{\boldsymbol{d}}_2,\ \underline{\boldsymbol{d}}_7\}$

$$\underline{\tilde{\boldsymbol{x}}}^2_B = \begin{bmatrix} \tilde{x}^2_1 \\ \tilde{x}^2_2 \\ \tilde{x}^2_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 0.25 \\ 0 \end{bmatrix}, \quad \underline{\tilde{\boldsymbol{x}}}^2_N = \tilde{x}^2_4 = 0$$

We note that both bases $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *represent the single extreme point or basic feasible solution* $\underline{\tilde{\boldsymbol{x}}} = [0.75,\ 0.25,\ 0,\ 0]$ *and this basic feasible solution is degenerate.*

From example (1.4.3) we note that a rank-deficient matrix A may lead to a degenerate basic feasible solution, we note that degeneracy of the basic feasible solution is not always the result of rank-deficiency of the underlying matrix, see for example Chapter 2 [50]. The following theorem gives another interpretation of degenerate basic feasible solutions.

Theorem 1.4.4. *[Theorem 3.4 [50]] For every extreme (basic feasible solution) there exists a corresponding basis, not necessarily unique and conversely, for every basis there exists a corresponding unique extreme point. Moreover, if an extreme point has more than one basis representing it, then it is degenerate.*

Degeneracy of a basic feasible solution causes computational difficulties in the Simplex algorithm. In the absence of degeneracy, phase (II) of the Simplex Algorithm moves from one basic feasible solution to another with an improved objective function value, then the algorithm either stops after a finite number of steps at an optimal solution or indicates that the linear program is unbounded. However, in the presence of degeneracy it is possible for a simplex direction in phase (II) to yield a step of length zero, which occurs as a result of the Simplex Algorithm "cycling" over a set of bases that all represent the same extreme point. Moreover, in some extreme examples the Simplex Algorihtm, e.g, see [82] the Simplex Algorithm fails to terminate as a result of "cycling" . It is however possible to prevent this "cycling" behaviour by augmenting the Simplex Algorithm and utilising Bland's rule in steps 1) and 3) of the Simplex Algorithm see Chapter 8 [51].

1.4.0.2 Dual Simplex Algorithm

Associated with every linear program is another linear program called its dual. The dual of the dual linear program is the original linear program, which is referred to as the primal linear program.

We recall that we are interested in solving the linear program (1.65), whose primal representation is given by

$$\begin{aligned} &\text{Minimise} \quad \underline{\boldsymbol{c}}^T \underline{\tilde{\boldsymbol{x}}} \\ &\text{subject to} \quad D\underline{\tilde{\boldsymbol{x}}} = \underline{\boldsymbol{b}} \\ &\qquad\qquad \underline{\tilde{\boldsymbol{x}}} \geq \underline{\mathbf{0}} \end{aligned}$$

where $D = [A, I] \in \mathbb{R}^{M\times(M+N)}$, $\underline{\tilde{\boldsymbol{x}}} = [\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}^a] \in \mathbb{R}^{(M+N)}$, $\underline{\boldsymbol{c}} = [0, \ldots, 0, 1, \ldots, 1]$. The corresponding dual of (1.65) is given by

$$\begin{aligned} &\text{Maximise} \quad \underline{\boldsymbol{b}}^T \underline{\boldsymbol{y}} \\ &\text{subject to} \quad D^T \underline{\boldsymbol{y}} \geq \underline{\boldsymbol{c}} \\ &\qquad\qquad \underline{\boldsymbol{y}} \in \mathbb{R}^M \quad \text{unrestricted} \end{aligned} \tag{1.67}$$

The dual is obtained from the primal, by following a sequence of rules transforming the minimisation problem into a maximisation, writing the right-hand side coefficients of the primal as the coefficients of the objective function of the dual, etc. For explicit rules of construction of the dual of any linear program see Chapter 9 [51].

The following theorem illustrates that as the Simplex Algorithm solves the primal problem, it also implicitly solves the dual problem, moreover it does so in such a way that the objective functions at optimal solution coincide.

Theorem 1.4.5. *[Theorem 5.2 [49]] Strong Duality Theorem*
If the primal problem with objective function $\underline{\boldsymbol{c}}^T \underline{\tilde{\boldsymbol{x}}}$ has an optimal solution $\underline{\tilde{\boldsymbol{x}}}^*$, then the corresponding dual problem with objective function $\underline{\boldsymbol{b}}^T \underline{\boldsymbol{y}}$ also has an optimal solution $\underline{\boldsymbol{y}}^*$, such that

$$\underline{\boldsymbol{c}}^T \underline{\tilde{\boldsymbol{x}}}^* = \underline{\boldsymbol{b}}^T \underline{\boldsymbol{y}}^*$$

We observe that (1.67) is not in standard form, that is it has inequality constraints. To transform (1.67) into standard form we need to add a slack vector $\underline{\boldsymbol{s}}$, thus obtaining

$$\begin{aligned} &\text{Maximise} \quad \underline{\boldsymbol{b}}^T \underline{\boldsymbol{y}} \\ &\text{subject to} \quad D^T \underline{\boldsymbol{y}} - \underline{\boldsymbol{s}} = \underline{\boldsymbol{c}} \\ &\qquad\qquad \underline{\boldsymbol{s}} \geq \underline{\mathbf{0}}, \quad \underline{\boldsymbol{y}} \quad \text{unrestricted} \end{aligned} \tag{1.68}$$

where $\underline{\boldsymbol{y}} \in \mathbb{R}^M$ and $\underline{\boldsymbol{s}} \in \mathbb{R}^{(M+N)}$.

When introducing slack variables to transform the linear problem into standard form, in either the primal or the dual linear problems we have the following theorem.

Theorem 1.4.6. *[Theorem 5.2 [49]] Complementary Slackness Theorem*
Suppose that $\underline{\tilde{\boldsymbol{x}}}^* = [\tilde{x}_1^*, \ldots, \tilde{x}_n^*]$ *is a feasible solution for the primal linear program and that* $\underline{\boldsymbol{y}}^* = [y_1^*, \ldots, y_m^*]$ *is feasible for the corresponding dual linear program. Let* $\underline{\boldsymbol{r}}^* = [r_1^*, \ldots, r_m^*]$ *denote the corresponding primal slack vector and let* $\underline{\boldsymbol{s}}^* = [s_1^*, \ldots, s_n^*]$ *denote the corresponding dual slack vector. Then* $\underline{\tilde{\boldsymbol{x}}}^*$ *and* $\underline{\boldsymbol{y}}^*$ *are optimal for their respective problems if and only if*

$$\tilde{x}_j^* s_j^* = 0 \quad \textit{for} \quad j \in \{1, \ldots, n\}$$

$$y_i^* r_i^* = 0 \quad \textit{for} \quad i \in \{1, \ldots, m\}$$

Similarly to the Primal Simplex Algorithm, we can formulate the Dual Simplex Algorithm which solves the primal linear program by implicitly solving its dual problem. This algorithm can also be formulated as a two-phase algorithm, where phase (I) consists of finding a dual basic feasible solution, which by complementary slackness, corresponds to a primal basic solution and phase (II) consists of moving from one basic feasible solution of the dual problem to an improved basic feasible solution until optimality of the dual, and thereby also the primal is reached. For explicit formulation of the Dual Simplex Algorithm see Chapter 11 [51].

The Dual Simplex Method is very important computationally, as it is typically faster than the Primal Simplex Method and has lower memory requirements in most optimisation software packages.

1.4.0.3 Primal-Dual Interior Point Method

Both Primal and Dual Simplex Algorithms rely on moving from one basic feasible solution to another along the edge of the feasible region. In the process, we always satisfy either primal or dual feasibility and complementary slackness and only satisfy all three conditions once the optimal solution has been found. The Primal-Dual Interior point method does neither of the above, in the sense that it moves through the interior of the feasible region and it does not always satisfy two of the aforementioned three conditions required for optimality. For a detailed discussion on the Primal-Dual Interior Point Method see Chapter 11.5 in [51].

Let us assume that, as before, the primal linear program is given by (1.65) and its dual is given by (1.68). By theorem (1.4.6) we know that $\underline{\tilde{\boldsymbol{x}}}^*$ is an optimal primal solution and $(\underline{\boldsymbol{y}}^*, \underline{\boldsymbol{s}}^*)$ is an optimal dual solution if and only if these solutions satisfy

$$D\underline{\tilde{\boldsymbol{x}}} = \underline{\boldsymbol{b}}, \quad \underline{\tilde{\boldsymbol{x}}} \geq \underline{\mathbf{0}}$$

$$D^T\underline{\boldsymbol{y}} - \underline{\boldsymbol{s}} = \underline{\boldsymbol{c}}, \quad \underline{\boldsymbol{s}} \geq \underline{\mathbf{0}}$$

$$\tilde{x}_j s_j = 0 \quad \text{for} \quad j \in \{1, \ldots, M+N\}$$

Assume that we have a strictly feasible solution $\underline{\tilde{\boldsymbol{x}}} > \underline{\mathbf{0}}$ to the primal problem; that is a feasible solution that is strictly greater than $\underline{\mathbf{0}}$ and a feasible solution $(\underline{\boldsymbol{y}}, \underline{\boldsymbol{s}})$ to the dual problem with $\underline{\boldsymbol{s}} > \underline{\mathbf{0}}$, thus satisfying

$$D\underline{\tilde{\boldsymbol{x}}} = \underline{\boldsymbol{b}}$$

$$D^T\underline{\boldsymbol{y}} - \underline{\boldsymbol{s}} = \underline{\boldsymbol{c}}$$

However, as these solutions are strictly feasible, complementary slackness condition does not hold. Thus we may restrict the solutions $\underline{\tilde{\boldsymbol{x}}}$ and $\underline{\boldsymbol{s}}$ to satisfy

$$\tilde{x}_j s_j = \mu > 0 \quad \text{for} \quad j \in \{1, \ldots, M+N\}$$

therefore we may interpret the solutions $\underline{\tilde{\boldsymbol{x}}}(\mu),\ \underline{\boldsymbol{s}}(\mu),\ \underline{\boldsymbol{y}}(\mu)$ as functions of μ and hence write

$$D\underline{\tilde{\boldsymbol{x}}}(\mu) = \underline{\boldsymbol{b}} \tag{1.69}$$

$$D^T\underline{\boldsymbol{y}}(\mu) - \underline{\boldsymbol{s}}(\mu) = \underline{\boldsymbol{c}} \tag{1.70}$$

$$\tilde{B}S\underline{\boldsymbol{e}} = \mu\underline{\boldsymbol{e}} \tag{1.71}$$

where $\tilde{B} = diag(\tilde{x}_1, \ldots, \tilde{x}_{M+N})$, $S = diag(s_1, \ldots, s_{M+N})$ and $\underline{\boldsymbol{e}} = [1, \ldots, 1]$, where (1.71) expresses the approximate the complementary slackness in matrix form. As $\mu \to 0$ any limiting solution to the aforementioned system of equations yields an optimal solution to the primal-dual pair.

However, finding feasible solutions to both the primal and the dual problems, that additionally satisfy (1.71) is difficult. Hence, we endeavour to approximate the solution to (1.71) . Supposing that for some $\mu > 0$, we have a solution $\left(\underline{\tilde{\boldsymbol{x}}}(\mu),\ \underline{\boldsymbol{s}}(\mu),\ \underline{\boldsymbol{y}}(\mu)\right)$ to (1.69) and (1.70), but the equation (1.71) is not necessarily satisfied, however we do have $\underline{\tilde{\boldsymbol{x}}}(\mu) > 0$ and $\underline{\boldsymbol{s}}(\mu) > 0$. Consequently, we may endeavour to construct directions $(\underline{\boldsymbol{d}}^{\tilde{\boldsymbol{x}}},\ \underline{\boldsymbol{d}}^{\boldsymbol{y}},\ \underline{\boldsymbol{d}}^{\boldsymbol{s}})$, such that $\underline{\tilde{\boldsymbol{x}}}(\mu) + \underline{\boldsymbol{d}}^{\tilde{\boldsymbol{x}}}$ satisfies (1.69), $\left(\underline{\boldsymbol{y}}(\mu) + \underline{\boldsymbol{d}}^{\boldsymbol{y}},\ \underline{\boldsymbol{s}}(\mu) + \underline{\boldsymbol{d}}^{\boldsymbol{s}}\right)$ satisfies (1.70) and $\left(\underline{\tilde{\boldsymbol{x}}}(\mu) + \underline{\boldsymbol{d}}^{\tilde{\boldsymbol{x}}},\ \underline{\boldsymbol{s}}(\mu) + \underline{\boldsymbol{d}}^{\boldsymbol{s}}\right)$ approximately satisfies (1.71).

Similarly to the construction of feasible directions for the Primal Simplex Method, in order for $\underline{\tilde{\boldsymbol{x}}}(\mu) + \underline{\boldsymbol{d}}^{\tilde{\boldsymbol{x}}}$ to satisfy (1.69) and $\left(\underline{\boldsymbol{y}}(\mu) + \underline{\boldsymbol{d}}^{\boldsymbol{y}},\ \underline{\boldsymbol{s}}(\mu) + \underline{\boldsymbol{d}}^{\boldsymbol{s}}\right)$ to satisfy (1.70) we require that

$$D\underline{\boldsymbol{d}}^{\tilde{\boldsymbol{x}}} = \underline{\boldsymbol{0}}$$

$$D^T\underline{\boldsymbol{d}}^{\boldsymbol{y}} - \underline{\boldsymbol{d}}^{\boldsymbol{s}} = \underline{\boldsymbol{0}}$$

Considering the equation for the approximate complementary slackness, we obtain

$$(\tilde{x}_j + d_j^{\tilde{\boldsymbol{x}}})(s_j + d_j^{\boldsymbol{s}}) = \mu$$

which can be approximated by solving

$$\tilde{x}_j d_j^{\boldsymbol{s}} + s_j d_j^{\tilde{\boldsymbol{x}}} = \mu - \tilde{x}_j s_j \tag{1.72}$$

Expressing (1.72) in matrix form, we obtain

$$\tilde{B}\underline{\boldsymbol{d}}^{\boldsymbol{s}} + S\underline{\boldsymbol{d}}^{\tilde{\boldsymbol{x}}} = \mu\underline{\boldsymbol{e}} - \tilde{B}S\underline{\boldsymbol{e}}$$

Therefore, for a fixed value of μ we can produce a new solution $\Big(\underline{\tilde{\boldsymbol{x}}}(\mu) + \underline{\boldsymbol{d}}^{\tilde{\boldsymbol{x}}},\ \underline{\boldsymbol{s}}(\mu) + \underline{\boldsymbol{d}}^{\boldsymbol{s}},\ \underline{\boldsymbol{y}}(\mu) + \underline{\boldsymbol{d}}^{\boldsymbol{y}}\Big)$, that solves (1.69) and (1.70) while better approximating (1.71), by solving

$$D\underline{\boldsymbol{d}}^{\tilde{\boldsymbol{x}}} = \underline{\mathbf{0}}$$

$$D^T\underline{\boldsymbol{d}}^{\boldsymbol{y}} - \underline{\boldsymbol{d}}^{\boldsymbol{s}} = \underline{\mathbf{0}}$$

$$\tilde{B}\underline{\boldsymbol{d}}^{\boldsymbol{s}} + S\underline{\boldsymbol{d}}^{\tilde{\boldsymbol{x}}} = \mu\underline{\boldsymbol{e}} - \tilde{B}S\underline{\boldsymbol{e}}$$

yielding

$$\underline{\boldsymbol{d}}^{\boldsymbol{y}} = \left(DS^{-1}\tilde{B}D^T\right)^{-1}DS^{-1}\underline{\boldsymbol{v}}(\mu)$$

$$\underline{\boldsymbol{d}}^{\boldsymbol{s}} = D^T\underline{\boldsymbol{d}}^{\boldsymbol{y}}$$

$$\underline{\boldsymbol{d}}^{\tilde{\boldsymbol{x}}} = S^{-1}\left(\underline{\boldsymbol{v}}(\mu) - \tilde{B}\underline{\boldsymbol{d}}^{\boldsymbol{s}}\right)$$

where $\underline{\boldsymbol{v}}(\mu) = \mu\underline{\boldsymbol{e}} - \tilde{B}S\underline{\boldsymbol{e}}$. We proceed to the next iteration by decreasing the value of μ until we have computed a predetermined number of steps or until the components $\tilde{x}_j s_j$ are close enough to zero. This approach constitutes the Primal-Dual Interior Point Method, for a full implementation of this method see Chapter 11 [51].

The Interior Point Method is also very important computationally, as opposed to the Primal and Dual Simplex Methods, it is often parallellized in optimisation software packages. Moreover, most software packages utilise the Interior Point Method as the default method for LP Problems, as it is usually faster than the Primal and Dual methods for large-scale problems. However, it has several major drawbacks: each iteration requires the solution of multiple systems of equations and rounding errors can become an issue; the algorithm starts with a strictly feasible solution to both the primal and the dual problem, which are not always easy to identify. Moreover, because $\mu > 0$, we do not typically end up at a basic feasible solution. However there are methods to generate a basic feasible solution from the optimal solution of the interior point method.

Chapter 2

Generalised Cubature Measures

Our construction of Carathéodory cubature measures can be generalised from Borel measures on Euclidean spaces to Radon measures on locally compact Hausdorff spaces. Moreover, the familiar notion of a cubature measure as given in definition (1.2.3), as a measure integrating exactly all elements of the basis of Π_m^d, $\mathcal{B}(\Pi_m^d) := \{\gamma_1, \ldots, \gamma_M\}$, can also be generalised. We can consider integrating any finite set of test functions $\Psi = \{\psi_1, \ldots, \psi_M\}$ of which $\mathcal{B}(\Pi_m^d) = \{\gamma_1, \ldots, \gamma_M\}$ is special case. These extensions of the notions introduced in Chapter 1 to a more general measure-theoretic framework are the scope of the following chapter.

The following proposition illustrates that a finite Borel measure on a metric space is uniquely determined by its values on closed sets, as well as open sets. In particular, the d-dimensional Lebesgue measure, for which we derived the construction of Carathéodory cubature measures in Chapter 1, was implicitly taken to have support on some compact subset of $\mathbb{R}^d$. Thus, the underlying Lebesgue measure was uniquely determined on the metric space $\mathbb{R}^d$ with standard Euclidean metric.

Proposition 2.0.7. *[Proposition 1.1.1 [76]] A finite Borel measure μ on a metric space X has the following properties:*

1) For any Borel set $B \subseteq X$ and any $\epsilon > 0$ there exists an open set G and a closed set F with $F \subseteq B \subseteq G$ such that

$$\mu(F \backslash G) \leq \epsilon$$

2)For any Borel set $B \subseteq X$

$$\mu(B) = sup\{\ \mu(F) \mid F\ closed \subseteq B\} = inf\{\ \mu(G) \mid G\ open \supseteq B\}$$

We recall the definition of a Hausdorff space.

Definition 2.0.8. *Let (X, T) be a topological space. We say that (X, T) is a Hausdorff space if*

$$\forall x, y \in X,\ x \neq y :\ \exists N_x,\ N_y \in T :\ x \in N_x,\ y \in N_y :\ N_x \cap N_y = \varnothing$$

Further, we recall that Borel measures on a Hausdorff space (X, T), are measures defined on $\mathcal{B}(X)$, the Borel σ-algebra $\mathcal{B}(X)$ generated by T.

As noted in remark(1.12) in [76], a Borel measure on a general Hausdorff space is also uniquely determined by its values on closed and open sets. However, (1) and (2) of proposition 2.0.7 need not hold. In order to have a good integration theory against a measure on a general Hausdorff space, we need to be able to approximate the measure of a set by measures of other sets within it. This and other considerations, lead us to consider Borel measures on Hausdorff spaces, that are finite on compact sets and inner regular, that is Radon measures, as defined in the following.

Definition 2.0.9. *A Borel measure μ on a Hausdorff space X is called a Radon measure if*
1) $\mu(K) < \infty$ for every compact set $K \subseteq X$
2) $\mu(B) = sup\{\ \mu(K) \mid K\ compact \subseteq B\ \}\ \forall B \in \mathcal{B}(X)$

When restricted to a locally compact Hausdorff space X, that is a Hausdorff space with the property that every point has a compact neighbourhood, Radon measures correspond to bounded linear functionals on the space of continuous, compactly supported functions on X, by Riesz Representation Theorem. As noted, in p.12 in [76], a Radon measure on locally compact Hausdorff space X is uniquely determined by its values on the family of compact subsets of X.

In the following, locally compact spaces will be assumed to be Hausdorff. Given a locally compact space X, let $M_+(X)$ denote the set of positive Radon measures on X.

Let X be a locally compact space and $\mu \in M_+(X)$. Among the open μ-null sets $G \subseteq X$ there exists a biggest. If $(G_i)_{i \in I}$ is the system of all open μ-null sets, then $G = \cup_{i \in I} G_i$ is an open μ-null set, therefore the biggest. To see that $\mu(G) = 0$, it is

enough to show that $\mu(K) = 0$, for an arbitrary compact $K \subseteq G$. By compactness of K, there exists a finite collection $I_0 \subseteq I$, such that $K \subseteq \cup_{i \in I_0} G_i$ and hence

$$\mu(K) \leq \sum_{i \in I_0} \mu(G_i) = 0$$

Definition 2.0.10. *[Definition 1.3.1 [76]]*
The support of μ, denoted by supp(μ), is the complement of the biggest open μ-null set.

Proposition 2.0.11. *[Proposition 1.3.2 [76]] Let $\mu \in M_+(X)$. Then supp(μ) is a closed set. For $x \in X$ we have $x \in supp(\mu) \Leftrightarrow \mu(N_x) > 0$ for every open neighbourhood N_x of x.*

We recall that a measure μ on $\mathbb{R}^N$ is said to have an absolute moment of order $n = (n_1, \ldots, n_N) \in \mathbb{N}_0^N$ if

$$\int_{\mathbb{R}^N} |\underline{\boldsymbol{x}}|^n d\mu(\underline{\boldsymbol{x}}) = \int_{\mathbb{R}^N} |x_1|^{n_1} \ldots |x_N|^{n_N} d\mu(\underline{\boldsymbol{x}}) < \infty$$

Let $\mathcal{M}_N^*$ denote the set of positive Radon measures on $\mathbb{R}^N$ with absolute moments of any order. That is, to say

$$\mathcal{M}_N^* = \left\{ \mu \in M_+(\mathbb{R}^N) \mid \int |\underline{\boldsymbol{x}}|^n d\mu(\underline{\boldsymbol{x}}) < \infty \quad \forall n \in \mathbb{N}_0^N \right\}$$

Then, we have the following

Theorem 2.0.12. *[Theorem 2.1.7 [76]]*
Every $\mu \in M_+(\mathbb{R}^N)$ with compact support belongs to $\mathcal{M}_N^*$.

Proof. Immediate from definition of a Radon measure □

Given a compact set $K \subset \mathbb{R}^N$, let $\mathcal{M}_N^*(K)$ denote the set of positive Radon measures on $\mathbb{R}^N$, supported on $K \subset \mathbb{R}^N$

$$\mathcal{M}_N^*(K) = \left\{ \mu \in M_+(\mathbb{R}^N) \mid supp(\mu) \subseteq K \right\}$$

In the following, we consider the extension of the notion of cubature measure to a measure integrating exactly any finite set of test functions $\{\psi_1, \ldots, \psi_N\}$.

Lemma 2.0.13. *Let X be a locally compact space and $\mu \in M_+(X)$ with $supp(\mu) \subseteq D$, $D \in \mathcal{B}(X)$ compact. Let $\Psi = (\psi_1, \dots, \psi_N) : D \to \mathbb{R}^N$ be a continuous map, then the push-forward measure of μ through Ψ is $\Psi_*\mu \in M_+(\mathbb{R}^N)$ with $supp(\Psi_*\mu) \subseteq \Psi(D)$, $\Psi(D) \in \mathcal{B}(\mathbb{R}^N)$ compact.*

Proof. Let $C(D)$ denote the space of continuous real-valued functions on the compact set D. By Riesz Representation Theorem, see Theorem 1.2.2 in [76], a positive Radon measure μ, concentrated on D, is identified with a unique positive bounded linear functional in $C(D)^*$

$$\mu \mapsto L_\mu(f) = \int_D f d\mu \qquad \text{for } f \in C(D)$$

where positivity of L_μ means that if $f \geq 0$, then $L_\mu(f) \geq 0$.
$\Psi(D) \subseteq \mathbb{R}^N$ is compact, since it is a continuous image of a compact set. Hence we can define two linear maps

$$\begin{aligned} \Psi^* : &\quad C(\Psi(D)) \to C(D) \quad \text{given by } f \mapsto \Psi^* f = f \circ \Psi \\ \Psi_* : &\quad M_+(X) \to M_+(\mathbb{R}^N) \quad \text{given by } \mu \mapsto \Psi_*\mu := \nu \end{aligned}$$

Noting that we can define Ψ_* , since Ψ is a continuous map between two topological Hausdorff spaces and hence is Borel measurable. Then

$$L_\nu(f) = \int_{\Psi(D)} f d\nu = \int_{\Psi(D)} f d(\Psi_*\mu) = \int_D \Psi^* f d\mu = L_\mu(\Psi^* f) \quad \text{for } f \in C(\Psi(D))$$

and $L_\nu(f) \geq 0$, whenever $f \geq 0$. Hence, by Riesz Representation Theorem there exists a unique measure $\nu = \Psi_*\mu \in M_+(\mathbb{R}^N)$, concentrated on $\Psi(D)$.

□

Definition 2.0.14. *Let X be a locally compact space, $\mu \in M_+(X)$ with $supp(\mu) \subseteq D$, $D \in \mathcal{B}(X)$. Let $\Psi = (\psi_1, \dots, \psi_N) : D \to \mathbb{R}^N$ be a continuous map. Then we call $\nu_\Psi^\mu \in M_+(X)$ a Ψ-generalised cubature measure for μ if*

1) Ψ is μ-integrable
2) $\int_X \psi_j(\underline{\boldsymbol{x}}) d\mu(\underline{\boldsymbol{x}}) = \int_X \psi_j(\underline{\boldsymbol{x}}) d\nu_\Psi^\mu(\underline{\boldsymbol{x}}) \quad \forall j \in [\![1, N]\!]$
3) $supp(\nu_\Psi^\mu) \subseteq supp(\mu)$

Lemma 2.0.15. *Let X be a locally compact space, $\mu \in M_+(X)$ with $supp(\mu) \subseteq D$, $D \in \mathcal{B}(X)$. Let $\Psi = (\psi_1, \dots, \psi_N) : D \to \mathbb{R}^N$ and $\Phi = (\phi_1, \dots, \phi_N) : D \to \mathbb{R}^N$*

be two continuous maps. If $\nu_{\Psi}^{\mu} \in M_+(X)$ is a Ψ-generalised cubature measure for μ and $span\{\psi_1, \ldots, \psi_N\} = span\{\phi_1, \ldots, \phi_N\}$, then there exists a Φ-generalised cubature measure for μ, $\nu_{\Phi}^{\mu} \in M_+(X)$.

Proof. If $span\{\psi_1, \ldots, \psi_N\} = span\{\phi_1, \ldots, \phi_N\}$, then for each $\underline{\boldsymbol{x}} \in D$, each $j \in [\![1, N]\!]$, $\exists \alpha_1^j, \ldots, \alpha_N^j \in \mathbb{R} : \phi_j(\underline{\boldsymbol{x}}) = \sum_{i=1}^N \alpha_i{}^j \psi_i(\underline{\boldsymbol{x}})$. If ν_{Ψ}^{μ} is a Ψ-generalised cubature measure for μ, clearly, Φ is μ-integrable, since is Ψ is. Then, we have

$$\int_X \phi_j(\underline{\boldsymbol{x}}) d\mu(\underline{\boldsymbol{x}}) = \sum_{i=1}^N \alpha_i{}^j \int_X \psi_j(\underline{\boldsymbol{x}}) d\nu(\underline{\boldsymbol{x}}) = \sum_{i=1}^N \alpha_i{}^j \int_X \psi_j(\underline{\boldsymbol{x}}) d\nu_{\Psi}^{\mu}(\underline{\boldsymbol{x}}) = \int_X \phi_j(\underline{\boldsymbol{x}}) d\nu_{\Psi}^{\mu}(\underline{\boldsymbol{x}})$$

and since $supp(\nu_{\Psi}^{\mu}) \subseteq supp(\mu)$ we can choose $\nu_{\Phi}^{\mu} = \nu_{\Psi}^{\mu}$. □

Definition 2.0.16. *Let X be a locally compact space, $\mu \in M_+(X)$ with $supp(\mu) \subseteq D$, $D \in \mathcal{B}(X)$. Let $\Psi = (\psi_1, \ldots, \psi_N) : D \to \mathbb{R}^N$ be a continuous map. Then we call $\widehat{\nu}_{\Psi}^{\mu} \in M_+(X)$ a Ψ-generalised Carathéodory cubature measure for μ if*

1) $\widehat{\nu}_{\Psi}^{\mu}$ is a Ψ-generalised cubature measure for μ
2) $card\big(supp(\widehat{\nu}_{\Psi}^{\mu})\big) \leq N + 1$

In order to show the existence of a Ψ-generalised Carathéodory cubature measure for $\mu \in M_+(X)$ with $supp(\mu) \subseteq D$, $D \in \mathcal{B}(X)$, D compact, we will need a preliminary result.

In the following, we denote the ring of real-valued d-variate polynomials of degree $\leq m \in \mathbb{N}$, by $\Pi_m^d = \mathbb{R}_m[X_1, \ldots, X_d]$.

Theorem 2.0.17. *([84] Theorem 1) Let μ be a positive, finite Borel measure on $\mathbb{R}^d$, with $supp(\mu) = K$, $K \in \mathcal{B}(\mathbb{R}^d)$ compact and $m \in \mathbb{N}$. Then there exists an atomic measure $\widehat{\mu}$ on $\mathbb{R}^d$, $\widehat{\mu} = \sum_{i=1}^{M'} \omega_i \delta_{\underline{\boldsymbol{x}}_i}$ with $\omega_i \in \mathbb{R}_+$, $\underline{\boldsymbol{x}}_i \in K$, $M' \leq M = dim(\Pi_m^d)$ such that*

$$\mathbb{E}_{\mu}[P] = \int_{\mathbb{R}^d} P(\underline{\boldsymbol{x}}) d\mu(\underline{\boldsymbol{x}}) = \sum_{i=1}^M \omega_i P(\underline{\boldsymbol{x}}_i) = \mathbb{E}_{\widehat{\mu}}[P]$$

$\forall P \in \Pi_m^d = \mathbb{R}_m[X_1, \ldots, X_d]$

The preceding theorem is an extension to Tchakaloff's Theorem to arbitrary positive Borel measures on $\mathbb{R}^d$. In [16], Tchakaloff proved the existence of an atomic measure $\widehat{\mu}$, given a positive, Borel, compactly supported measure μ, that is absolutely continuous with respect to the d-dimensional Lebesgue measure.

Corollary 2.0.18. *Let $\mu \in M_+(X)$ with $supp(\mu) \subseteq D$, $D \in \mathcal{B}(X)$ compact. Given a continuous, μ-integrable map $\Psi = (\psi_1, \ldots, \psi_N) : D \to \mathbb{R}^N$, there exists a Ψ-generalised Carathéodory cubature measure $\widehat{\nu}_\Psi^\mu$ for μ*

Proof. Let us define two linear maps

$$\begin{aligned} \Psi^* :&\ \ C(\Psi(D)) \to C(D) \quad \text{given by } f \mapsto \Psi^* f = f \circ \Psi \\ \Psi_* :&\ \ M_+(X) \to M_+(\mathbb{R}^N) \quad \text{given by } \mu \mapsto \Psi_* \mu := \nu \end{aligned}$$

Then, by lemma 2.0.13, ν is a positive Radon measure on $\mathbb{R}^N$, concentrated on the compact set $\Psi(D)$. And, in particular $\nu \in M_N^*(\Psi(D))$, hence

$$\int_{\mathbb{R}^N} \|\underline{\boldsymbol{z}}\| d\nu(\underline{\boldsymbol{z}}) = \int_X \|\Psi(\underline{\boldsymbol{x}})\| d\mu(\underline{\boldsymbol{x}}) < \infty$$

By Theorem 2.0.17, there exists an atomic measure $\widehat{\nu}$ on $\mathbb{R}^N$, $\widehat{\nu} = \sum_{i=1}^R w_i \delta_{\underline{\boldsymbol{z}}_i}$ with $\omega_i \in \mathbb{R}_+,\ \underline{\boldsymbol{z}}_i \in \Psi(D),\ R \leq dim(\Pi_N^1) = N+1$ such that

$$\int_{\mathbb{R}^N} P(\underline{\boldsymbol{z}}) d\nu(\underline{\boldsymbol{z}}) = \sum_{i=1}^R \omega_i P(\underline{\boldsymbol{z}}_i)$$

$\forall P \in \Pi_N^1 = \mathbb{R}[Z_1, \ldots, Z_N]$
Since

$$\int_{\mathbb{R}^N} P(\underline{\boldsymbol{z}}) d\nu(\underline{\boldsymbol{z}}) = \int_{\mathbb{R}^N} P(\underline{\boldsymbol{z}}) d(\Psi^*\mu)(\underline{\boldsymbol{z}}) = \int_X \Psi^* P(\underline{\boldsymbol{x}}) d\mu(\underline{\boldsymbol{x}}) = \sum_{i=1}^R \omega_i P(\Psi(\underline{\boldsymbol{x}}_i))$$

we have $\Psi(\underline{\boldsymbol{x}}_i) = z_i$ and thus we obtain an atomic measure $\widehat{\nu}_\Psi^\mu$ on X, $\widehat{\nu}_\Psi^\mu = \sum_{i=1}^R w_i \delta_{\underline{\boldsymbol{x}}_i}$ with $\omega_i \in \mathbb{R}_+,\ \underline{\boldsymbol{x}}_i \in D$. In particular, choosing $P_j(\psi_1(x), \ldots, \psi_N(x)) = \psi_j(x)$, we get

$$\int_X \psi_j(\underline{\boldsymbol{x}}) d\mu(\underline{\boldsymbol{x}}) = \sum_{i=1}^R \omega_i \psi_j(\underline{\boldsymbol{x}}_i) := \int_X \psi_j(\underline{\boldsymbol{x}}) d\widehat{\nu}_\Psi^\mu(\underline{\boldsymbol{x}}) \quad \forall j \in [\![1, N]\!]$$

Clearly $\widehat{\nu}_\Psi^\mu \in M_+(X)$ with $supp(\widehat{\nu}_\Psi^\mu) \subset supp(\mu)$ and $card\big(supp(\widehat{\nu}_\Psi^\mu)\big) = R \leq N+1$ $\square$

In order to prove the existence of a Ψ-generalised Carathéodory cubature measure for a positive Radon measure μ, supported on an arbitrary Borel set $B \subseteq X$, we need a preliminary result.

Theorem 2.0.19. *([2] Theorem 1, Corollary 2) Let μ be a positive Borel measure on $\mathbb{R}^d$, with $supp(\mu) \subseteq B$, $B \in \mathcal{B}(\mathbb{R}^d)$, admitting the first absolute moment*

$$\int_{\mathbb{R}^d} \|\underline{\boldsymbol{x}}\| d\mu(\underline{\boldsymbol{x}}) < \infty$$

Then, the first moment of μ is contained in $cone(B)$, that is $\int_{\mathbb{R}^d} \underline{\boldsymbol{x}} d\mu(\underline{\boldsymbol{x}}) \subseteq cone(B)$. And there exists an atomic measure measure $\widehat{\mu}$ on $\mathbb{R}^d$, $\widehat{\mu} = \sum_{i=1}^{R} \omega_i \delta_{\underline{\boldsymbol{x}}_i}$ with $\omega_i \in \mathbb{R}_+$, $\underline{\boldsymbol{x}}_i \in B$, $R \leq dim(\Pi_1^d)$ such that

$$\mathbb{E}_\mu[P] = \int_{\mathbb{R}^d} P(\underline{\boldsymbol{x}}) d\mu(\underline{\boldsymbol{x}}) = \sum_{i=1}^{R} \omega_i P(\underline{\boldsymbol{x}}_i) = \mathbb{E}_{\widehat{\mu}}[P]$$

$\forall P \in \Pi_1^d = \mathbb{R}[X_1, \ldots, X_d]$

Corollary 2.0.20. *Let $\mu \in M_+(X)$ with $supp(\mu) \subseteq D$, $D \in \mathcal{B}(X)$. Given a continuous, μ-integrable map $\Psi = (\psi_1, \ldots, \psi_N) : D \to \mathbb{R}^N$. Assume that $(\Psi_* \mu)$ admits the first absolute moment*

$$\int_{\mathbb{R}^N} \|\underline{\boldsymbol{x}}\| d(\Psi_* \mu)(\underline{\boldsymbol{x}}) < \infty$$

Then, there exists a Ψ-generalised Carathéodory cubature measure $\widehat{\nu}_\Psi^\mu$ for μ

Proof. Set $\nu := \Psi_* \mu$, then ν is a positive Radon measure on $\mathbb{R}^N$, $supp(\nu) \subseteq \Psi(D)$, $\Psi(D) \in \mathcal{B}(\mathbb{R}^N)$. By Theorem 2.0.19 there exists an atomic measure $\widehat{\nu}$ on $\mathbb{R}^N$, $\widehat{\nu} = \sum_{i=1}^{R} w_i \delta_{\underline{\boldsymbol{z}}_i}$ with $\omega_i \in \mathbb{R}_+$, $\underline{\boldsymbol{z}}_i \in \Psi(D)$, $R \leq dim(\Pi_N^1) = N + 1$ such that

$$\int_{\mathbb{R}^N} P(\underline{\boldsymbol{z}}) d\nu(\underline{\boldsymbol{z}}) = \sum_{i=1}^{R} \omega_i P(\underline{\boldsymbol{z}}_i)$$

$\forall P \in \Pi_N^1 = \mathbb{R}[Z_1, \ldots, Z_N]$

We proceed similarly to the proof of Corollary 2.0.18, to obtain an atomic measure $\widehat{\nu}_\Psi^\mu \in M_+(X)$, which is the Ψ-generalised cubature measure for μ. □

Definition 2.0.21. *Let $d, m \in \mathbb{N}$ be given and let $\mathcal{B}(\Pi_m^d) = \{\gamma_1, \ldots, \gamma_M\}$ denote some fixed basis for Π_m^d, with $\gamma_1(\underline{\boldsymbol{x}}) = 1$. Let $\left(\Gamma_m^d[\mathbb{R}],\ \leq_{grlex}\right)$ denote the ordered collection of monomials of degree $\leq m$ on $\mathbb{R}^d$, that is $\Gamma_m^d[\mathbb{R}] = \{\gamma_j(\underline{\boldsymbol{x}})\}_{|j| \leq m}$ where the ordering in $\Gamma_m^d[\mathbb{R}]$ is given by the graded lexicographical order, which is a total order defined by*

$$\gamma_i(\underline{\boldsymbol{x}}) <_{grlex} \gamma_k(\underline{\boldsymbol{x}}) \Leftrightarrow \Big(deg(\gamma_i(\underline{\boldsymbol{x}})) < deg(\gamma_k(\underline{\boldsymbol{x}}))\Big) \vee \Big(deg(\gamma_i(\underline{\boldsymbol{x}})) = deg(\gamma_k(\underline{\boldsymbol{x}})) \wedge i <_{lex} k\Big)$$

for $\gamma_i, \gamma_k \in \Gamma_m^d[\mathbb{R}]$

Then, $P \in \Pi_m^d$ are identified with linear maps on $\Gamma_m^d[\mathbb{R}]$, $P(\underline{\boldsymbol{x}}) = \sum_{|k|=0}^{\left|\Gamma_m^d[\mathbb{R}]\right|} \alpha_k \gamma_k(\underline{\boldsymbol{x}})$. Equipped with notion of an ordered set of monomials in $\mathbb{R}^d$, we can define the (m, d)-*monomial form* of a general point $\underline{\boldsymbol{x}} \in \mathbb{R}^d$.

Definition 2.0.22. *Let $d, m \in \mathbb{N}$ be given. Let $M = \left|\Gamma_m^d[\mathbb{R}]\right|$ and define the (m, d)-monomial mapping $\gamma_m^d : \mathbb{R}^d \to \mathbb{R}^M$ by*

$$\gamma_m^d(\underline{\boldsymbol{x}}) = \left[\gamma_1(\underline{\boldsymbol{x}}), \gamma_2(\underline{\boldsymbol{x}}), \ldots, \gamma_M(\underline{\boldsymbol{x}})\right]^T \qquad \gamma_k \in \Gamma_m^d[\mathbb{R}]$$

where $\gamma_1(\underline{\boldsymbol{x}}) \equiv 1$, $\forall \underline{\boldsymbol{x}} \in \mathbb{R}^d$, and the enumeration of γ_k is taken according to $\leq_{grlex}$. We refer to $\gamma_m^d(\underline{\boldsymbol{x}})$ as the (m, d)-monomial form of $\underline{\boldsymbol{x}} \in \mathbb{R}^d$.

Definition 2.0.23. *Let $\mu \in M_+(\mathbb{R}^d)$ with $supp(\mu) \subseteq B$, $B \in \mathcal{B}(\mathbb{R}^d)$. Then we call the push-forward measure of μ through γ_m^d, $\gamma_{m*}^d\mu$ the (m, d)-monomial form of μ.*

Definition 2.0.24. *Let $d, m \in \mathbb{N}$ be given and $\mu \in M_+(\mathbb{R}^d)$ with $supp(\mu) \subseteq D$, $D \in \mathcal{B}(\mathbb{R}^d)$. Let $M = \left|\Gamma_m^d[\mathbb{R}]\right|$ and let $\gamma_m^d : D \to \mathbb{R}^M$ denote the (m, d)-monomial mapping. Then we call $\nu_m^\mu \in M_+(\mathbb{R}^d)$, the cubature measure of degree m for μ if*

1) γ_m^d is μ-integrable
2) $\int_{\mathbb{R}^d} \gamma_j(\underline{\boldsymbol{x}}) d\mu(\underline{\boldsymbol{x}}) = \int_{\mathbb{R}^d} \gamma_j(\underline{\boldsymbol{x}}) d\nu_m^\mu(\underline{\boldsymbol{x}}) \quad \forall j \in [\![1, M]\!]$
3) $supp(\nu_m^\mu) \subseteq supp(\mu)$

Definition 2.0.25. *Let $d, m \in \mathbb{N}$ be given and $\mu \in M_+(\mathbb{R}^d)$ with $supp(\mu) \subseteq D$, $D \in \mathcal{B}(\mathbb{R}^d)$, and let $M = \left|\Gamma_m^d[\mathbb{R}]\right|$. Then we call $\widehat{\nu}_m^\mu \in M_+(\mathbb{R}^d)$, Carathéodory cubature measure of degree m for μ if*
1) $\widehat{\nu}_m^\mu$ is a cubature measure of degree m for μ
2)$card(supp(\widehat{\nu}_m^\mu)) \leq M + 1$

Corollary 2.0.26.
Let $d, m \in \mathbb{N}$ be given and $\mu \in M_+(\mathbb{R}^d)$ with $supp(\mu) \subseteq D$, $D \in \mathcal{B}(\mathbb{R}^d)$. Let $\gamma_m^d : D \to \mathbb{R}^M$ denote the (m, d)-monomial mapping, then
1) If D is compact, there exists Carathéodory cubature measure of degree m for μ
2) If D is an arbitrary Borel set and

$$\int_{\mathbb{R}^M} \|\underline{\boldsymbol{x}}\| d(\gamma_{m*}^d \mu)(\underline{\boldsymbol{x}}) < \infty$$

there exists Carathéodory cubature measure of degree m for μ

Proof. By choosing $\Psi = \gamma_m^d$ and applying corollaries (2.0.18) and (2.0.20) respectively, we obtain results 1) and 2). □

Chapter 3

Recombination Algorithm

As outlined in Section 1.3.1, the Recombination Algorithm can be utilised to produce Carathéodory cubatures $Q^{CAR}(m,d)$ from the Cartesian tensor product cubature $Q(m,d)$ for moderate degrees $m \in \mathbb{N}$ and dimensions $d \in \mathbb{N}$. However, prior to specialising the Recombination Algorithm to the computation of Carathéodory cubatures, we introduce the abstract framework of the Recombination Algorithm. This is the scope of the forthcoming chapter.

Furthermore, we shall introduce a novel algorithm based on Singular Value Decomposition to perform the computationally dominant part of the Recombination Algorithm, that is computing a basic feasible solution $\underline{\widehat{\boldsymbol{\beta}}} = [\beta_1, \ldots, \beta_{M'}, 0, \ldots, 0]^T \in \mathbb{R}^L_+$, with $M' \leq M$ and $L \geq M+1$ to

$$\begin{aligned} &A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}, \qquad A \in \mathbb{R}^{M\times L}, \quad \underline{\boldsymbol{x}} \in \mathbb{R}^L_+, \quad \underline{\boldsymbol{b}} \in \mathbb{R}^M \\ &\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}, \ \sum_{i=1}^{L} x_i = 1 \end{aligned} \tag{3.1}$$

provided that we already have a strictly feasible solution to the system (3.1), $\underline{\boldsymbol{\beta}} = [\beta_1, \ldots, \beta_L]^T \in \mathbb{R}^L_+$, satisfying $A\underline{\boldsymbol{\beta}} = \underline{\boldsymbol{b}}$.

This new SVD-based algorithm will serve as replacement for the Simplex Algorithm in the implementation of the Recombination Algorithm, and as we shall illustrate in Chapter 5 the SVD-based algorithm improves the overall computational runtime of Carathéodory cubature measure constructions by one order of magnitude.

Let $\nu \in \mathcal{M}_+(\mathbb{R}^d)$ with $supp(\nu) \subseteq K$, $K \in \mathcal{B}(\mathbb{R}^d)$ compact be given by

$$\nu = \sum_{i=1}^{N} \omega_i \delta_{\underline{\boldsymbol{x}}_i} \qquad \omega_i \geq 0,\ \underline{\boldsymbol{x}}_i \in K$$

Without loss of generality, we assume that ν is a probability measure, if it is not, we set $\nu = \nu'$

$$\nu' = \sum_{i=1}^{N} \omega_i' \delta_{\underline{\boldsymbol{x}}_i} \qquad \text{where} \quad \omega_i' = \frac{\omega_i}{\sum_{i=1}^{N} \omega_i}$$

Let $\Psi = (\psi_1, \ldots, \psi_M) : \ K \to \mathbb{R}^M$ be a continuos, ν-integrable map. Suppose that we wish to construct a Ψ-generalised Carathéodory cubature measure for ν, defined in (2.0.16), that is a measure $\widehat{\nu}_\Psi \in \mathcal{M}_+(\mathbb{R}^d)$, satisfying

$$\begin{aligned}
&1)\ \int_K \psi_j(\underline{\boldsymbol{x}}) d\nu(\underline{\boldsymbol{x}}) = \int_K \psi_j(\underline{\boldsymbol{x}}) d\widehat{\nu}_\Psi(\underline{\boldsymbol{x}}) \quad \forall j \in [\![1, M]\!] \qquad (3.2)\\
&2)\ supp(\widehat{\nu}_\Psi) \subseteq supp(\nu)\\
&3)\ card\big(supp(\widehat{\nu}_\Psi^\mu)\big) \leq M + 1
\end{aligned}$$

We can define the push-forward of ν through Ψ by

$$(\Psi_* \nu) = \sum_{i=1}^{N} \omega_i \delta_{\Psi(\underline{\boldsymbol{x}}_i)} \qquad \omega_i \geq 0,\ \Psi(\underline{\boldsymbol{x}}_i) \in \Psi(K) \subseteq \mathbb{R}^M$$

Then, clearly $(\Psi_* \nu)$ is a probability measure on $\mathbb{R}^M$ and we have

$$\begin{aligned}
\mathcal{C}oM\big(\Psi_* \nu\big) &= \Big[\sum_{i=1}^{N} \omega_i \psi_1(\underline{\boldsymbol{x}}_i), \ldots, \sum_{i=1}^{N} \omega_i \psi_M(\underline{\boldsymbol{x}}_i) \Big]^T\\
&= \Big[\int_K \psi_1(\underline{\boldsymbol{x}}) d\nu, \ldots, \int_K \psi_M(\underline{\boldsymbol{x}}) d\nu \Big]^T
\end{aligned}$$

where given an atomic measure $\mu = \sum_{i=1}^{N} \lambda_i \delta_{\underline{\boldsymbol{x}}_i}$ the centre of mass of μ is defined as $\mathcal{C}oM(\mu) \triangleq \sum_{i=1}^{N} \lambda_i \underline{\boldsymbol{x}}_i$.

Let us now introduce the notion of a reduced measure for an atomic probability measure on $\mathbb{R}^M$.

Definition 3.0.27 (Definition 1 [96]). *Let $L, M \in \mathbb{N}$ be such that $L \geq M+1$. Let $\mu \in \mathcal{M}_+(\mathbb{R}^M)$, $supp(\mu) \subseteq K$, $K \in \mathcal{B}(\mathbb{R}^M)$ compact, $\mu(K) = 1$, $card(supp(\mu)) = L$. Then, we call the probability measure $\tilde{\mu}_{M'} \in \mathcal{M}_+(\mathbb{R}^M)$, a reduced measure for μ if*

$$\begin{aligned} &1) \quad \mathcal{C}oM(\tilde{\mu}_{M'}) = \mathcal{C}oM(\mu) \\ &2) \quad supp(\tilde{\mu}_{M'}) \subset supp(\mu) \\ &3) \quad card(supp(\tilde{\mu}_{M'})) = M^{'} \leq M \end{aligned}$$

Lemma 3.0.28. *Let $L, M \in \mathbb{N}$ be such that $L \geq M+1$. Let $\mu \in \mathcal{M}_+(\mathbb{R}^M)$, $supp(\mu) \subseteq K$, $K \in \mathcal{B}(\mathbb{R}^M)$ compact, $\mu(K) = 1$, $card(supp(\mu)) = L$. Then, there exists a probability measure $\tilde{\mu}_{M'} \in \mathcal{M}_+(\mathbb{R}^M)$, that is a reduced measure for μ.*

Proof. Apply Carathéodory Convex Hull Theorem.

□

Remark 3.0.29. *Given $L, M \in \mathbb{N}$ with $L \geq M+1$, the reduced measure $\tilde{\mu}_{M'}$ for μ,in definition (3.0.27), is generally not unique. Indeed as will be illustrated in the forthcoming section, the number of reduced measures $\tilde{\mu}_{M'}$ for μ, is upper bounded by $\binom{L}{M}$.*

Consider setting the probability measure $\mu := (\Psi_*\nu) \in \mathcal{M}_+(\mathbb{R}^M)$, $supp(\Psi_*\nu) \subseteq \Psi(K)$ compact, $\Psi(K) \in \mathcal{B}(\mathbb{R}^M)$. Then, by lemma (3.0.28) there exists a reduced measure for $(\Psi_*\nu)$, given by

$$\tilde{\mu}_{M'} := \widehat{\Psi_*\nu} = \sum_{k=1}^{M^{'}} \widehat{\omega}_k \delta_{\Psi(\underline{\boldsymbol{x}}_{i_k})} \qquad \widehat{\omega}_k \geq 0, \ \sum_{k=1}^{M^{'}} \widehat{\omega}_k = 1, \quad M^{'} \leq M$$

satisfying

$$\sum_{k=1}^{M^{'}} \widehat{\omega}_k \Psi(\underline{\boldsymbol{x}}_{i_k}) = \mathcal{C}oM(\widehat{\Psi_*\nu}) = \mathcal{C}oM(\Psi_*\nu) = \sum_{i=1}^{N} \widehat{\omega}_i \Psi(\underline{\boldsymbol{x}}_i) \tag{3.3}$$

with $\{\Psi(\underline{\boldsymbol{x}}_{i_1}), \ldots, \Psi(\underline{\boldsymbol{x}}_{i_{M'}})\} \subset \{\Psi(\underline{\boldsymbol{x}}_1), \ldots, \Psi(\underline{\boldsymbol{x}}_N)\}$. Thus, yielding an atomic probability measure $\widehat{\nu}$ on $\mathbb{R}^d$

$$\widehat{\nu} = \sum_{k=1}^{M^{'}} \widehat{\omega}_k \delta_{\underline{\boldsymbol{x}}_{i_k}}$$

satisfying condition 1) in (3.19) by (3.3) and satisfying conditions 2) and 3) in (3.19), by construction. Hence, by construction we obtain a Ψ-generalised Carathéodory cubature measure for ν and we set $\widehat{\nu}_\Psi = \widehat{\nu}$.

However, in order to construct the reduced measure $\tilde{\mu}_{M'} := (\widehat{\Psi_*\nu})$ for $\mu := (\Psi_*\nu)$ and thereby obtain Ψ-generalised Carathéodory cubature measure for ν, efficiently, we will employ an iterative procedure. This procedure can be summarised as follows

For $\big(j \geq 1\big)$ Construct a probability measure $\widehat{\mu}^j \in \mathcal{M}_+(\mathbb{R}^M)$, satisfying

$$\begin{aligned} &1)\ \mathcal{C}oM(\widehat{\mu}^j) = \mathcal{C}oM(\mu) \\ &2)\ supp(\widehat{\mu}^j) \subset supp(\widehat{\mu}^{j-1}) \subset supp(\mu) \\ &3)\ card(supp(\widehat{\mu}^j)) < card(supp(\widehat{\mu}^{j-1})) < card(supp(\mu)) \end{aligned} \tag{3.4}$$

If $\big(card(supp(\widehat{\mu}^j)) \leq M\big)$ terminate and output $\tilde{\mu}_{M'} := \widehat{\mu}^j$

We note that, by definition, each measure $\widehat{\mu}^j$ gives rise to $\widehat{\nu}^j$, a Ψ-generalised cubature measure for ν, as conditions 1) and 2) in (3.19) are satisfied by conditions 1) and 2) in the construction of $\widehat{\mu}^j$. Whence, we obtain a sequence $\{\widehat{\nu}^j\}_j$ of Ψ-generalised cubature measures for ν and we terminate when $\widehat{\nu}^j$ satisfies condition 3) in (3.19), thereby yielding a Carathéodory cubature measure for ν.

In order to construct a sequence of measures $\widehat{\mu}^j$ in (3.4), satisfying the aforementioned properties, we proceed to cluster the weights and nodes of $(\Psi_*\nu)$ in such a way, so as to preserve its centre of mass, whilst producing a small number of clusters for faster elimination by the forthcoming measure reduction algorithm. Moreover, a further priority in this construction, is to subdivide each cluster in the same centre of mass preserving fashion. This will enable us to introduce a recursive procedure on these clusters, relying on cluster reduction and subdivision, terminating with a subset of the original nodes and weights of $(\Psi_*\nu)$.

3.1 Recursive Halving Forest Construction Algorithm

In the following we shall present a hierarchical clustering algorithm, building on the ideas introduced in section 1.3.3. This algorithm introduces a binary forest data storage structure for cubature nodes and weights, which allows us to utilise recursion for cubature measure reduction, by descending align the forest of data trees generated. We will call this algorithm RHFC (Recursive Halving Forest Construction) Algorithm. This clustering Algorithm takes its roots in the "Recursive Doubling Algorithm", primarily utilised in parallel computing paradigms for efficient message passing between computing nodes, see [53] and also for efficient parallel matrix computations [52].

A slight variant of the RHFC Algorithm is presented in [96]. The first difference between the RHFC Algorithm and the algorithm presented in [96], is the emphasis on the binary forest data representation of the hierarchical clustering structure of cubature nodes and weights. The second difference is as follows. In [96], given a cubature problem of some degree m and dimension d, thus with underlying space of polynomials of dimension $M := dim(\Pi_m^d)$, the cubature data is aways partitioned into $(M+1)$ clusters. In the present work, given a cubature problem of some degree m and dimension d, we illustrate how to partition the cubature data into any collection of clusters L, where $M+1 \leq L < N$. In particular, we illustrate that within the broader context of the Recombination Algorithm, the hierarchical clustering into $2M$ clusters yields a one order of magnitude speed-up (from $O(M^3) \rightarrow O(M^4)$) when contrasted with $(M+1)$ cluster collection.

We shall assume, for simplicity of exposition, that $N = 2^S$, for some $S \in \mathbb{N}$. Initially, we populate the leaf nodes of the forthcoming binary tree data structure with weights and nodes of $(\Psi_* \nu)$

$$(\omega_i^0, \underline{\boldsymbol{x}}_i^0) := \Big(\omega_i, \Psi(\underline{\boldsymbol{x}}_i)\Big) \qquad\qquad i \in [\![1, 2^S]\!]$$

this will denote the zeroth level of the binary tree T_*. We fill the jth level with $k = 2^{S-j}$ nodes, consisting of pairs $(\omega_k^j, \underline{\boldsymbol{x}}_k^j)$, which can be computed recursively by taking the following convex combinations of the leaf nodes

Iterate through $j = 1 : S$, setting $k = 1 : 2^{S-j},\ i = 2k - 1$

$$\underline{\boldsymbol{x}}_k^j := \frac{\left(\omega_i^{j-1}\underline{\boldsymbol{x}}_i^{j-1} + \omega_{i+1}^{j-1}\underline{\boldsymbol{x}}_{i+1}^{j-1}\right)}{\left(\omega_i^{j-1} + \omega_{i+1}^{j-1}\right)} \tag{3.5}$$

each node $\underline{\boldsymbol{x}}_k^j$ on level j is a convex combination of the two constituent children nodes $\underline{\boldsymbol{x}}_i^{j-1},\ \underline{\boldsymbol{x}}_{i+1}^{j-1}$ on level $j-1$.

Moreover, expanding the expression for the children nodes recursively we obtain

$$\underline{\boldsymbol{x}}_k^j := \frac{\sum_{r=(k-1)2^j+1}^{k2^j} \omega_r^0 \underline{\boldsymbol{x}}_r^0}{\sum_{r=(k-1)2^j+1}^{k2^j} \omega_r^0} = \sum_{r=(k-1)2^j+1}^{k2^j} \omega_r^{'} \underline{\boldsymbol{x}}_r^0 \tag{3.6}$$

$$\text{where} \quad \omega_r^{'} = \frac{\omega_r^0}{\sum_{r=(k-1)2^j+1}^{k2^j} \omega_r^0}, \qquad \sum_{r=(k-1)2^j+1}^{k2^j} \omega_r^{'} = 1$$

Hence, each node $\underline{\boldsymbol{x}}_k^j$ is a convex combination over $\left\{\underline{\boldsymbol{x}}_r^0\right\}_{r=(k-1)2^j+1}^{k2^j}$, which is a sub-collection of 2^j leaf nodes.

Therefore, for $1 \le j \le S,\ 1 \le k \le 2^{S-j}$, we have

$$\underline{\boldsymbol{x}}_k^j \in Conv\Big(\{\underline{\boldsymbol{x}}_1^0, \ldots, \underline{\boldsymbol{x}}_N^0\}\Big)$$

We form the weights associated with node $\underline{\boldsymbol{x}}_k^j$, by adding the weights of the two constituent children nodes on level $j-1$

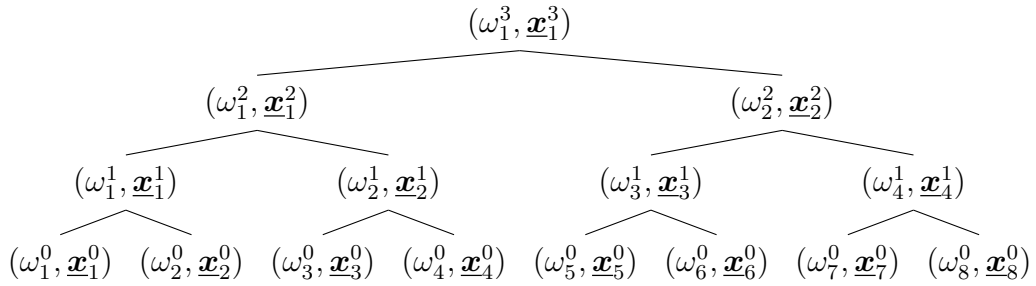
$$\omega_k^j := \left(\omega_i^{j-1} + \omega_{i+1}^{j-1}\right) = \sum_{r=(k-1)2^j+1}^{k2^j} \omega_r^0 \tag{3.7}$$

which results in computing the total mass over the sub-collection $\left\{\underline{\boldsymbol{x}}_r^0\right\}_{r=(k-1)2^j+1}^{k2^j}$ of leaf nodes.

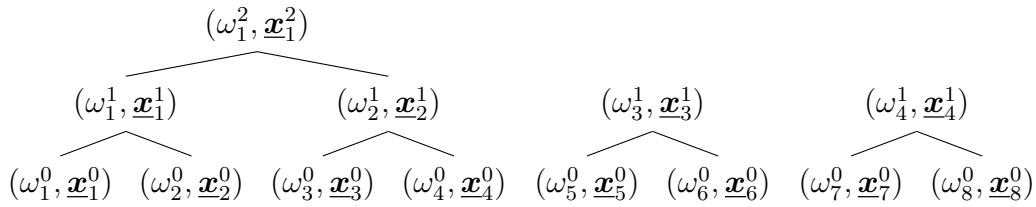
As shown by (3.6) and (3.7) it is readily verified, that for each level $j \in [\![1, S]\!]$, we have

$$\sum_{k=1}^{2^{S-j}} \omega_k^j \underline{\boldsymbol{x}}_k^j = \sum_{i=1}^{2^S} \omega_i^0 \underline{\boldsymbol{x}}_i^0 = \mathcal{C}oM(\Psi_* \nu)$$

By employing the aforementioned iterative procedure, we build a full binary tree T_* of depth S. The figure below illustrates T_* with $N = 2^3$.



For any $L \in [\![1, 2^S]\!]$, we can construct a binary forest $\{T_1, \ldots, T_L\}$ of full binary trees, by trimming the desired number of nodes from the top levels of T_*. The resulting collection of binary trees will constitute our clusters. By construction of T_* we observe that any collection of trees, obtained by the process of trimming a certain number of nodes from the top levels of T_* will generate a binary forest, whose roots nodes preserving the centre of mass of the leaf nodes, as illustrated in the following example. Let us trim the root node $(\omega_1^3, \underline{\boldsymbol{x}}_1^3)$ and the right-hand node on the penultimate level $(\omega_2^2, \underline{\boldsymbol{x}}_2^2)$, of T_*, yielding the binary forest



We obtain a binary forest $\{T_1, T_2, T_3\}$ with root nodes

$$(\omega_1^2, \underline{\boldsymbol{x}}_1^2) = \left(\sum_{i=1}^{4} \omega_i^0, \ \frac{\sum_{i=1}^{4} \omega_i^0 \underline{\boldsymbol{x}}_i}{\sum_{i=1}^{4} \omega_i^0} \right)$$

$$(\omega_3^1, \underline{\boldsymbol{x}}_3^1) = \left(\sum_{i=5}^{6} \omega_i^0, \ \frac{\sum_{i=5}^{6} \omega_i^0 \underline{\boldsymbol{x}}_i}{\sum_{i=5}^{6} \omega_i^0} \right)$$

$$(\omega_4^1, \underline{\boldsymbol{x}}_4^1) = \left(\sum_{i=7}^{8} \omega_i^0, \ \frac{\sum_{i=7}^{8} \omega_i^0 \underline{\boldsymbol{x}}_i}{\sum_{i=7}^{8} \omega_i^0} \right)$$

It is readily verified that setting $\beta_1 := \omega_1^2,\ \beta_2 := \omega_1^3,\ \beta_3 := \omega_1^4$ and $\underline{\boldsymbol{y}}_1 := \underline{\boldsymbol{x}}_1^2,\ \underline{\boldsymbol{y}}_2 := \underline{\boldsymbol{x}}_3^1,\ \underline{\boldsymbol{y}}_3 := \underline{\boldsymbol{x}}_4^1$ gives rise to a measure $\mathcal{C}oM(\eta_3) = \sum_{i=1}^3 \beta_i \underline{\boldsymbol{y}}_i$, supported on the root nodes of a binary forest $\{T_1, T_2, T_3\}$, whose leaf nodes are points in the $supp(\Psi_*\nu)$ satisfying

$$\mathcal{C}oM(\eta_3) = \sum_{i=1}^{3} \beta_i \underline{\boldsymbol{y}}_i = \sum_{i=1}^{8} \omega_i^0 \underline{\boldsymbol{x}}_i^0 = \mathcal{C}oM(\Psi_*\nu)$$

$$\sum_{k=1}^{2^{3-j}} \omega_k^j \underline{\boldsymbol{x}}_k^j = \sum_{i=1}^{8} \omega_i^0 \underline{\boldsymbol{x}}_i^0 = \mathcal{C}oM(\Psi_*\nu)$$

The only constraint, that we impose on the binary forest $\{T_1, \ldots, T_L\}$, resulting from trimming T_* is that the depth(T_i) for $1 \leq i \leq L$ does not differ by more than 1. This keeps the underlying clusters nearly the same size, thereby allowing us to better manage the overall workload of the forthcoming algorithm.

By dropping the assumption $N = 2^S$, we can apply the RHFC (Recursive Halving Forest Construction) algorithm, detailed below, to $(\Psi_*\nu)$. However, the collection of trees $\{T_1, \ldots, T_L\}$ generated by RHFC, with $N \neq 2^S$ will not consist of perfect, but of complete, binary trees. We recall the definition of a complete binary tree.

Definition 3.1.1. *A binary tree $\{T,\ T^\ell,\ T^r\}$ of depth $D \geq 0$, where T^ℓ and T^r represent the left and right subtrees of T respectively, is complete if*
1) If $D = 0,\ T^\ell = \varnothing,\ T^r = \varnothing$

2) If $D > 0$, then either: T^ℓ is a perfect binary tree (every node is a leaf or possesses exactly two children) of depth $(D-1)$ and T^r is tree of depth $(D-1)$, which has every level completely full except possibly the last one with all its nodes to the left side; or
T^ℓ is a binary tree of height $(D-1)$, which has every level completely full except possibly the last one with all its nodes to the left side and T^r is a perfect binary tree of depth $(D-2)$.

Algorithm 3: Recursive Halving Forest Construction

Input: Positive atomic measure: $\mu = \sum_{i=1}^{N} \lambda_i \delta_{\underline{\boldsymbol{z}}_i}$, Number of trees: $L \geq M+1$

1) Set $(\omega_i^0, \underline{\boldsymbol{x}}_i^0) := (\omega_i, \underline{\boldsymbol{z}}_i)$ for $i \in [\![1, N]\!]$ and $D := \lceil log_2\big(\frac{N}{L}\big) \rceil$

2) **For** $j = 1 : D$

If $\big(j = D, \lceil \frac{N}{2^D} \rceil < L\big)$

Set $r := \lceil \frac{N}{2^{D-1}} \rceil - L$

For $k = 1 : r,\ i = 2k-1$

If $\big(\lceil \frac{N}{2^{D-1}} \rceil = 2\ell + 1,\ \ell \in \mathbb{N}\big)$

$i = i - 1$

If $(k = 1)$

Set $(\omega_1^D, \underline{\boldsymbol{x}}_1^D) :=$

$$\left(\big(\omega_1^{D-1} + \omega^{D-1}_{\lceil \frac{N}{2^{D-1}} \rceil}\big), \frac{\big(\omega_1^{D-1}\underline{\boldsymbol{x}}_1^{D-1} + \omega^{D-1}_{\lceil \frac{N}{2^{D-1}} \rceil}\underline{\boldsymbol{x}}^{D-1}_{\lceil \frac{N}{2^{D-1}} \rceil}\big)}{\big(\omega_1^{D-1} + \omega^{D-1}_{\lceil \frac{N}{2^{D-1}} \rceil}\big)} \right)$$

Set $(\omega_k^D, \underline{\boldsymbol{x}}_k^D) :=$

$$\left((\omega_i^{D-1} + \omega_{i+1}^{D-1}), \frac{(\omega_i^{D-1}\underline{\boldsymbol{x}}_i^{D-1} + \omega_{i+1}^{D-1}\underline{\boldsymbol{x}}_{i+1}^{D-1})}{(\omega_i^{D-1} + \omega_{i+1}^{D-1})} \right)$$

end k **loop**

Else

For $k = 1 : \lceil \frac{N}{2^j} \rceil,\ i = 2k-1$

If $\big(\lceil \frac{N}{2^{j-1}} \rceil = 2\ell + 1,\ \ \ell \in \mathbb{N}\big)$

$i = i - 1$

If $(k = 1, j \neq 1\ )$

Set $(\omega_1^j, \underline{\boldsymbol{x}}_1^j) :=$

$$\left(\big(\omega_1^{j-1} + \omega^{j-1}_{\lceil \frac{N}{2^{j-1}} \rceil}\big), \frac{\big(\omega_1^{j-1}\underline{\boldsymbol{x}}_1^{j-1} + \omega^{j-1}_{\lceil \frac{N}{2^{j-1}} \rceil}\underline{\boldsymbol{x}}^{j-1}_{\lceil \frac{N}{2^{j-1}} \rceil}\big)}{\big(\omega_1^{j-1} + \omega^{j-1}_{\lceil \frac{N}{2^{j-1}} \rceil}\big)} \right)$$

If $\left(k = \lceil \frac{N}{2^j} \rceil\right)$

Set $(\omega_k^j, \underline{\boldsymbol{x}}_k^j) := (\omega_i^{j-1}, \underline{\boldsymbol{x}}_i^{j-1})$

Set $(\omega_k^j, \underline{\boldsymbol{x}}_k^j) :=$

$$\left(\left(\omega_i^{j-1} + \omega_{i+1}^{j-1}\right), \frac{\left(\omega_i^{j-1}\underline{\boldsymbol{x}}_i^{j-1} + \omega_{i+1}^{j-1}\underline{\boldsymbol{x}}_{i+1}^{j-1}\right)}{\left(\omega_i^{j-1} + \omega_{i+1}^{j-1}\right)} \right)$$

end k **loop**

end j **loop**

3) **If** $\left(\lceil \frac{N}{2^{D-1}} \rceil = 2\ell + 1,\ \ell \in \mathbb{N}\right)$

Set $p := 2r$

Else

Set $p := 2r + 1$

For $i = 1 : L$

If $(i \leq r)$

Set $(\beta_i, \underline{\boldsymbol{y}}_i) := (\omega_i^D, \underline{\boldsymbol{x}}_i^D)$

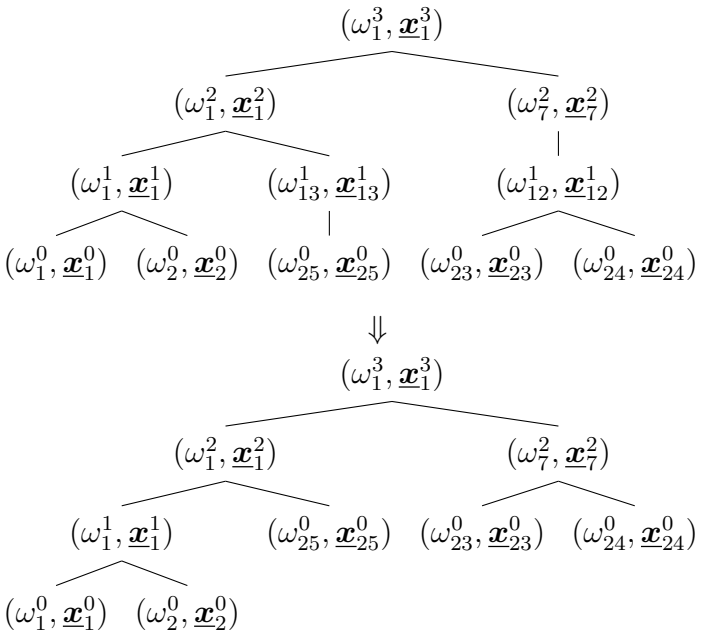
Else

Set $(\beta_i, \underline{\boldsymbol{y}}_i) := (\omega_{p+i-(r+1)}^{D-1}, \underline{\boldsymbol{x}}_{p+i-(r+1)}^{D-1})$

end i **loop**

Output: Atomic measure: $\eta_L = \sum_{i=1}^{L} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$

Under the above construction, when the number of nodes on a given level is odd, the last node on the successive level is a parent to only one child, which in turn corresponds to renaming the child node. If we were to drop all the straight edges connecting parent to a single child, thereby keeping just the children nodes, the resulting binary tree would always be complete. An illustration is given below for a single tree in the binary forest of $L = 5$ trees with an initial population of $N = 25$.



Remark 3.1.2. *We note that the output measure from the RHFC algorithm,* $\eta_L = \sum_{i=1}^{L} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$ *contains the information about the underlying binary forest structure. To make this dependence explicit, we say that the pair* $(\beta_i, \underline{\boldsymbol{y}}_i)$ *is a root pair of some complete binary tree* T_i *of depth* $depth(T_i) = D$ *or* $D-1$ *and it consists of the root node* $\underline{\boldsymbol{y}}_i = R(T_i)$ *and associated root weight* $\beta_i = P(T_i)$

$$\eta_L = \sum_{i=1}^{L} \beta_i \delta_{\underline{\boldsymbol{y}}_i}, \quad \beta_i = P(T_i), \quad \underline{\boldsymbol{y}}_i = R(T_i) \quad for \quad 1 \leq i \leq L$$

3.1.0.4 L-Tree forms

Definition 3.1.3. *Let $N, L \in \mathbb{N}$ be such that $N > L$. Let $\mu \in \mathcal{M}_+(\mathbb{R}^M)$, with $supp(\mu) \subseteq K$, $K \in \mathcal{B}(\mathbb{R}^M)$ compact, be given by*

$$\mu = \sum_{i=1}^{N} \lambda_i \delta_{\underline{\boldsymbol{z}}_i}$$

Then, we call $\eta_L \in \mathcal{M}_+(\mathbb{R}^M)$, with $supp(\eta_L) \subseteq Conv\big(\{\underline{\boldsymbol{z}}_1, \ldots, \underline{\boldsymbol{z}}_N\}\big) \subseteq K$, the L-tree form of μ, if

$$\eta_L = \sum_{i=1}^{L} \beta_i \delta_{\underline{\boldsymbol{y}}_i}, \quad \beta_i = P(T_i), \quad \underline{\boldsymbol{y}}_i = R(T_i) \quad for \quad 1 \leq i \leq L$$

satisfies

$$\mathcal{C}oM(\eta_L) = \mathcal{C}oM(\mu)$$

where $\{T_1, \ldots, T_L\}$ denotes a forest of complete binary trees, such that $depth(T_i) = J$ for $1 \leq i \leq s$ and $depth(T_i) = J-1$ for $s+1 \leq i \leq L$ with $0 \leq J \leq D$, $D = \lceil log_2\big(\frac{N}{L}\big) \rceil$ with root nodes $R(T_i)$

$$R(T_i) = \begin{cases} \underline{\boldsymbol{x}}_i^J & for \quad 1 \leq i \leq s \\ \underline{\boldsymbol{x}}_i^{J-1} & for \quad s+1 \leq i \leq L \end{cases} \tag{3.8}$$

and associated root weights $P(T_i)$

$$P(T_i) = \begin{cases} \omega_i^J & for \quad 1 \leq i \leq s \\ \omega_i^{J-1} & for \quad s+1 \leq i \leq L \end{cases} \tag{3.9}$$

and each level $j \in [\![0, J]\!]$ of the tree T_i contains node pairs

$$(\omega_k^j, \underline{\boldsymbol{x}}_k^j) \qquad for \quad 1 \leq k \leq L2^{J-j} \quad or \quad L2^{(J-1)-j}$$

satisfying the recursion

$$\omega_k^j := \big(\omega_{2k-1}^{j-1} + \omega_{2k}^{j-1}\big)$$

$$\underline{\boldsymbol{x}}_k^j := \frac{\big(\omega_{2k-1}^{j-1} \underline{\boldsymbol{x}}_{2k-1}^{j-1} + \omega_{2k}^{j-1} \underline{\boldsymbol{x}}_{2k}^{j-1}\big)}{\big(\omega_{2k-1}^{j-1} + \omega_{2k}^{j-1}\big)}$$

and the root nodes $\underline{\boldsymbol{x}}_i^J$, is produced by an application of the RHFC Algorithm to μ.

Remark 3.1.4. *We note that in the above definition the L-tree form of μ, η_L is supported on a forest of complete binary trees of some depth J or $J-1$, hence differing by one level at most.*

If depth $J = D$, then the resulting forest of binary trees is the output produced by the RHFC Algorithm applied to μ. However, for $J < D$, the resulting collection of binary trees is such that the nodes are constructed by the RHFC Algorithm, but the weights are not. Thus, we can regard the collection of trees in the support of η_L as some re-weighted sub-collection of sub-trees, produced by the RHFC Algorithm. This definition, will become more transparent in the following sections, as we illustrate the explicit construction of different L-tree forms of μ.

Lemma 3.1.5. *Let $N, L \in \mathbb{N}$ be such that $N > L$. Let $\mu \in \mathcal{M}_+(\mathbb{R}^M)$, with $supp(\mu) \subseteq K$, $K \in \mathcal{B}(\mathbb{R}^M)$ compact, $card(supp(\mu)) = N$ and let $\eta_L \in \mathcal{M}_+(\mathbb{R}^M)$ be an L-tree form of μ, supported on a complete binary forest $\{T_1, \ldots, T_L\}$, with each level $j \in [\![0, J]\!]$ of the tree T_i containing node pairs*

$$(\omega_k^j, \underline{\boldsymbol{x}}_k^j) \qquad \text{for} \quad 1 \leq k \leq L2^{J-j} \quad \text{or} \quad L2^{(J-1)-j}$$

Then,

$$\sum_k \omega_k^j \underline{\boldsymbol{x}}_k^j = \mathcal{C}oM(\mu), \qquad \text{for each level } j \in [\![0, J]\!]$$

Proof. By definition of an L-tree form of μ, we have $\mathcal{C}oM(\eta_L) = \mathcal{C}oM(\mu)$. Hence by expanding $\mathcal{C}oM(\eta_L)$ in terms of the expressions (3.8) and (3.9) and exploitng the recursive definition of the node pairs $(\omega_k^j, \underline{\boldsymbol{x}}_k^j)$, we obtain the desired result. □

Lemma 3.1.6. *Let $N, L \in \mathbb{N}$ be such that $N > L$. Let $\mu \in \mathcal{M}_+(\mathbb{R}^M)$, with $supp(\mu) \subseteq K$, $K \in \mathcal{B}(\mathbb{R}^M)$ compact, $card(supp(\mu)) = N$ and let $\eta_L \in \mathcal{M}_+(\mathbb{R}^M)$ be an L-tree form of μ, supported on a complete binary forest $\{T_1, \ldots, T_L\}$, with each level $j \in [\![0, J]\!]$ of the tree T_i containing node pairs*

$$(\omega_k^j, \underline{\boldsymbol{x}}_k^j) \qquad \text{for} \quad 1 \leq k \leq L2^{J-j} \quad \text{or} \quad L2^{(J-1)-j}$$

the following are satisfied

$$1)\ \bigcup_{i=1}^{L} l(T_i) \subseteq supp(\mu)$$

where $l(T_i)$ denotes the collection of leaf nodes of tree T_i

$$2) P(\widehat{T}_i)R(\widehat{T}_i) = P(\widehat{T}_i^{\ell})R(\widehat{T}_i^{\ell}) + P(\widehat{T}_i^{r})R(\widehat{T}_i^{r})$$

where $\widehat{T}_i$ denotes any subtree of T_i, including T_i itself and $\widehat{T}_i^{\ell}, \widehat{T}_i^{r}$ denote the left and right subtrees of $\widehat{T}_i$ of depth $\big(depth(T_i) - 1\big)$, respectively.

Proof. Property 1) holds true by construction of the RHFC Algorithm, if $J = D$, η_L is the output measure, produced by the RHFC Algorithm applied to μ, then we have

$$\bigcup_{i=1}^{L} l(T_i) = supp(\mu)$$

If $J < D$, then the collection of binary trees $\{T_1, \ldots, T_L\}$ is some re-weighted sub-collection of the binary forest produced by the RHFC Algorithm and hence we have

$$\bigcup_{i=1}^{L} l(T_i) \subset supp(\mu)$$

Property 2) holds true by the recursive expressions for nodes and weights in definition (3.1.3). Since for any level j,we have

$$\begin{aligned} P(\widehat{T}_i)R(\widehat{T}_i) = \omega_k^j \underline{\boldsymbol{x}}_k^j &= (\omega_i^{j-1} + \omega_{i+1}^{j-1}) \left(\frac{\omega_i^{j-1} \underline{\boldsymbol{x}}_i^{j-1} + \omega_{i+1}^{j-1} \underline{\boldsymbol{x}}_{i+1}^{j-1}}{\omega_i^{j-1} + \omega_{i+1}^{j-1}} \right) \\ &= \omega_i^{j-1} \underline{\boldsymbol{x}}_i^{j-1} + \omega_{i+1}^{j-1} \underline{\boldsymbol{x}}_{i+1}^{j-1} \\ &= P(\widehat{T}_i^{\ell})R(\widehat{T}_i^{\ell}) + P(\widehat{T}_i^{r})R(\widehat{T}_i^{r}) \end{aligned}$$

□

Remark 3.1.7. *Given an input $\mu \in \mathcal{M}_+(\mathbb{R}^M)$ into the RHFC Algorithm, given by*

$$\mu = \sum_{i=1}^{N} \lambda_i \delta_{\underline{\boldsymbol{z}}_i}$$

we have already remarked that the output measure η_L produced by the RHFC Algorithm is an L-tree form of μ.

This can be easily verified, since η_L satisfies the following properties

$$\eta_L = \sum_{i=1}^{L} \beta_i \delta_{\underline{\boldsymbol{y}}_i}, \quad \beta_i = P(T_i), \quad \underline{\boldsymbol{y}}_i = R(T_i)$$

where each resulting tree T_i is a complete binary tree of depth $J = D$ or $J = D-1$, where $D = \lceil log_2\big(\frac{N}{L}\big)\rceil$ and $s = r = \Big(\lceil \frac{N}{2^{D-1}} \rceil - L\Big)$

$$P(T_i) = \begin{cases} \omega_i^D & for \quad 1 \le i \le r \\ \omega_{p+i-(r+1)}^{D-1} & for \quad r+1 \le i \le L \end{cases} \qquad R(T_i) = \begin{cases} \underline{\boldsymbol{x}}_i^D & for \quad 1 \le i \le r \\ \underline{\boldsymbol{x}}_{p+i-(r+1)}^{D-1} & for \quad r+1 \le i \le L \end{cases}$$

and

$$\begin{aligned} \mathcal{C}oM(\eta_L) &= \sum_{i=1}^{L} \beta_i \underline{\boldsymbol{y}}_i \\ &= \sum_{i=1}^{r} \omega_i^D \underline{\boldsymbol{x}}_i^D + \sum_{i=r+1}^{L} \omega_{p+i-(r+1)}^{D-1} \underline{\boldsymbol{x}}_{p+i-(r+1)}^{D-1} \\ &= \sum_{i=1}^{r} \Big(\sum_{\ell=(i-1)2^D+1}^{i2^D} \lambda_\ell \underline{\boldsymbol{z}}_\ell \Big) + \sum_{i=r+1}^{L} \Big(\sum_{\ell=(r+i-1)2^{D-1}+1}^{(r+i)2^{D-1}} \lambda_\ell \underline{\boldsymbol{z}}_\ell \Big) \\ &= \sum_{\ell=1}^{N} \lambda_\ell \underline{\boldsymbol{z}}_\ell = \mathcal{C}oM(\mu) \end{aligned}$$

where the penultimate equality is obtained by expanding the expressions in (3.6) *and* (3.7).

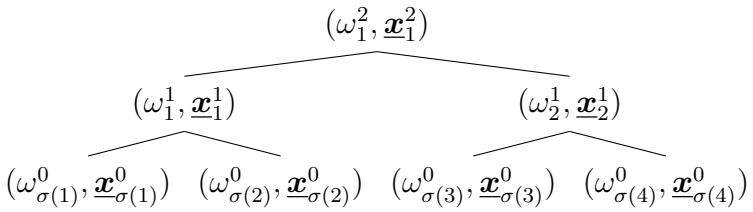
And for each level $j \in [\![1, J]\!]$, the recursive definition of the children nodes $(\omega_k^j, \underline{\boldsymbol{x}}_k^j)$ holds by construction. Moreover, for each level $j \in [\![1, J]\!]$, we have

$$\sum_k \omega_k^j \underline{\boldsymbol{x}}_k^j = \mathcal{C}oM(\mu)$$

which again holds by expanding the expressions for $(\omega_i^D, \underline{\boldsymbol{x}}_i^D)$ and $(\omega_{p+i-(r+1)}^{D-1}, \underline{\boldsymbol{x}}_{p+i-(r+1)}^{D-1})$ using (3.6) *and* (3.7).

3.1.0.5 Tree Permutations

Clearly, given an input $\mu \in \mathcal{M}_+(\mathbb{R}^M)$ into the RHFC Algorithm, the $\underline{L\text{-tree form of } \mu}$, η_L produced by the RHFC Algorithm is not unique, since any permutation of the leaf pairs of nodes and weights yields a potentially different output measure η_L . Consider, for example



for any $\sigma \in S_4$, where and S_4 denotes the symmetric group of degree 4. For example $\sigma_1 = (1)(2)(3)(4)$ and $\sigma_2 = (12)(34)$ yield the same $\underline{L\text{-tree form of } \mu}$,

$$\omega_1^1 = \omega_1^0 + \omega_2^0, \ \underline{\boldsymbol{x}}_1^1 = \underline{\boldsymbol{x}}_0^1 + \underline{\boldsymbol{x}}_0^2$$

$$\omega_2^1 = \omega_3^0 + \omega_4^0, \ \underline{\boldsymbol{x}}_2^1 = \underline{\boldsymbol{x}}_0^3 + \underline{\boldsymbol{x}}_0^4$$

$$\omega_1^2 = \omega_1^1 + \omega_2^1, \ \underline{\boldsymbol{x}}_1^2 = \underline{\boldsymbol{x}}_1^1 + \underline{\boldsymbol{x}}_2^1$$

similarly $\sigma_3 = (123)(4)$ and $\sigma_4 = (134)(2)$ yield the same $\underline{L\text{-tree form of } \mu}$. In total there up to $|S_N|/2 = N!/2$ different $\underline{L\text{-tree forms of } \mu}$ for even N, produced by the RHFC Algorithm.

We note that, since the overall goal of the RHFC Algorithm is to cluster the weights and nodes $(\omega_i^0, \underline{\boldsymbol{x}}_i^0)$ of the input measure μ into a complete binary forest of trees, differing by at most one in depth, whilst satisfying the two following properties

$$1)\ \mathcal{C}oM(\eta_L) = \mathcal{C}oM(\mu)$$

$$2)\ \sum_k \omega_k^j \underline{\boldsymbol{x}}_k^j = \mathcal{C}oM(\mu), \qquad \text{for each level } j \in [\![0, D]\!]$$

we do not make any distinction in the quality of the output measures produced by permuting the leaf nodes, as for any permutation $\sigma \in S_N$ of the leaf nodes the measure η_L produced by the RHFC, will satisfy these properties.

3.1.0.6 Reduced Measure

We recall that our goal is to construct a sequence of measures $\widehat{\mu}^j \in \mathcal{M}_+(\mathbb{R}^M)$, satisfying (3.4), where $\mu := (\Psi_*\nu)$. To accomplish this task, we have derived a methodology to construct the L-Tree form of μ, supported on the collection of binary trees $\{T_1, \ldots, T_L\}$, with the property that the union of its leaf nodes form the support of the original measure μ.

In the following, we shall couple the definition of a reduced measure with the idea of tree re-weighting, which will be introduced in Section 3.1.0.7. This will enable us to formulate the notion of a reduced tree measure $\eta_{M'}$ for η_L, supported on some subset of the binary forest $\{T_1, \ldots, T_L\}$. Subsequently, we shall illustrate that the leaf nodes of $\eta_{M'}$ give rise to $\widehat{\mu}^j$, thereby satisfying (3.4).

Applying the RHFC Algorithm to $(\Psi_*\nu)$, given by

$$(\Psi_*\nu) = \sum_{i=1}^{N} \omega_i \delta_{\Psi(\underline{\boldsymbol{x}}_i)} \qquad \omega_i \geq 0,\ \sum_{i=1}^{N} \omega_i = 1,\ \Psi(\underline{\boldsymbol{x}}_i) \in \mathbb{R}^M$$

yields an $\underline{L\text{-tree form of } (\Psi_*\nu)}$, given by

$$\eta_L = \sum_{i=1}^{L} \beta_i \delta_{\underline{\boldsymbol{y}}_i}, \quad \beta_i = P(T_i) \geq 0,\ \sum_{i=1}^{L} \beta_i = 1, \quad \underline{\boldsymbol{y}}_i = R(T_i) \in Conv\big(\{\Psi(\underline{\boldsymbol{x}}_1), \ldots, \Psi(\underline{\boldsymbol{x}}_N)\}\big)$$

such that

$$\mathcal{C}oM(\Psi_*\nu) = \sum_{i=1}^{N} \omega_i \Psi(\underline{\boldsymbol{x}}_i) = \sum_{i=1}^{L} \beta_i \underline{\boldsymbol{y}}_i = \mathcal{C}oM(\eta_L)$$

By lemma (3.0.28), we know that there exists reduced measure $\tilde{\eta}_{M'}$ for η_L with $card(supp(\tilde{\eta}_{M'})) = M^{'} \leq M$, which we can denote by

$$\tilde{\eta}_{M'} = \sum_{i=1}^{M^{'}} \widehat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}}$$

Since we have $supp(\tilde{\eta}_{M'}) \subset supp(\eta_L) = \{\underline{\boldsymbol{y}}_1, \ldots, \underline{\boldsymbol{y}}_L\} = \big\{R(T_1), \ldots, R(T_L)\big\}$, there must exist a binary forest $\{T_{i_1}, \ldots, T_{i_{M'}}\} \subset \{T_1, \ldots, T_L\}$ such that

$$\underline{\boldsymbol{y}}_{i_k} = R(T_{i_k}) \quad \text{for} \quad 1 \leq i \leq M^{'}$$

and hence we conclude that constructing $\tilde{\eta}_{M'}$ entails eliminating $(L - M)$ binary trees from the forest $\{T_1, \ldots, T_L\}$.

We note that by construction of $\tilde{\eta}_{M'}$, not only do the root nodes $\underline{\boldsymbol{y}}_{i_k}$ of T_{i_k} remain unaltered, but all the children nodes down each tree $T_{i_k} \in \{T_{i_1}, \ldots, T_{i_M}\}$ also remain unaltered.

However, it is only the root weights of the reduced measure $\tilde{\eta}_{M'}$ that are updated from the weights of η_L, by the following

$$\beta_k = P(T_{i_k}) \to \widehat{\beta}_k \quad \text{for} \quad 1 \leq k \leq M^{'}, \qquad \sum_{k=1}^{M^{'}} \widehat{\beta}_k = 1$$

Since, we are concerned with reducing the support of $(\Psi_* \nu)$ and by the construction of η_L, we know that the leaf nodes $\{\ell(T_1), \ldots, \ell(T_L)\}$ of the binary forest $\{T_1, \ldots, T_L\}$ in the $supp(\eta_L)$, satisfy

$$\bigcup_{i=1}^{L} l(T_i) = supp(\Psi_* \nu)$$

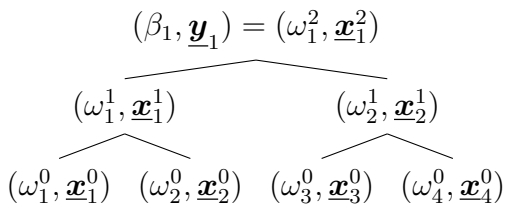
We can conclude that the leaf nodes of the reduced measure $\eta_{M'}$ satisfy

$$\bigcup_{k=1}^{M^{'}} l(T_{i_k}) \subset supp(\Psi_* \nu)$$

In the following section, we will illustrate that the construction of the reduced measure $\tilde{\eta}_{M'}$, enables us to reduce the support of $(\Psi_* \nu)$ in a centre of mass preserving fashion. To accomplish this, we need to permeate the root weight update of each tree T_{i_k}, $\beta_k \to \widehat{\beta}_k$ all the way down each level of the tree T_{i_k} and hence re-weight the leaf level of every remaining tree.

3.1.0.7 Updating Binary Tree Weights

In order to perform the tree weight update operation efficiently, we note the following: assume, for illustration $T_{i_1} = T_1 \in \{T_1, \ldots, T_L\}$ and depth(T_{i_1})=2, then T_{i_1} is given by



By construction of the RHFC Algorithm, we know that

$$\beta_1 := \omega_1^2 = \omega_1^1 + \omega_2^1$$

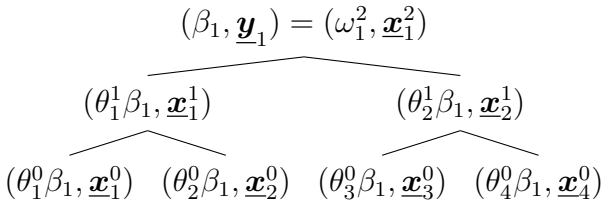
$$\beta_1 := \omega_1^2 = \omega_1^0 + \omega_2^0 + \omega_3^0 + \omega_4^0$$

since we need to keep track of each child node ω_k^j as a proportion of the root node $\beta_1 := \omega_1^2$ to perform updates, we let $\theta_k^j = (\omega_k^j / \beta_1)$ and hence we have

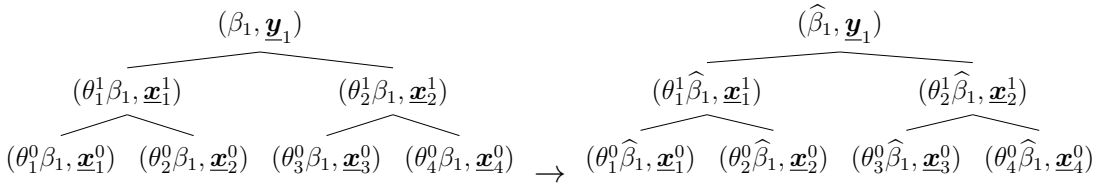
$$\beta_1 = \theta_1^1 \beta_1 + \theta_2^1 \beta_1$$

$$\beta_1 = \theta_1^0 \beta_1 + \theta_2^0 \beta_1 + \theta_3^0 \beta_1 + \theta_4^0 \beta_1$$

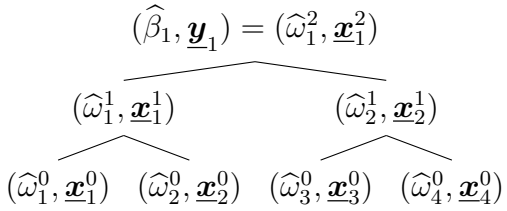
Therefore, instead of storing each individual child node ω_k^j in the tree structure above, we can store the product $(\theta_k^j \beta_1)$, resulting in



Whence, to permeate the update of the root weight of T_{i_1}, that is $\beta_1 \to \widehat{\beta}_1$, to all the children nodes of T_{i_1}, we perform the following update



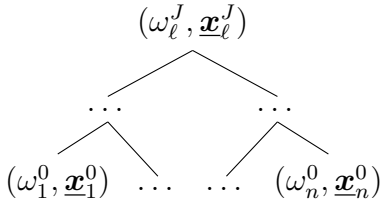
We denote the resulting, re-weighted tree T_{i_1} by



where $\widehat{\omega}_k^j = \theta_k^j\widehat{\beta}_1$.

If performed sequentially, the tree weight updates require touching every node of the tree T_{i_1}, hence at most $2^{depth(T_{i_1})} - 1$ nodes. However, we note that, this type of data storage for tree weights allows for $O(1)$ tree weight updates $\theta_k^j\beta_1 \rightarrow \theta_k^j\widehat{\beta}_1$, when performed in parallel.

Definition 3.1.8. *Let T be a complete binary tree of depth $J \in \mathbb{N}$*



with the root pair $(\omega_\ell^J, \underline{\boldsymbol{x}}_\ell^J)$ and each level $j \in [\![0, J]\!]$ contains node pairs

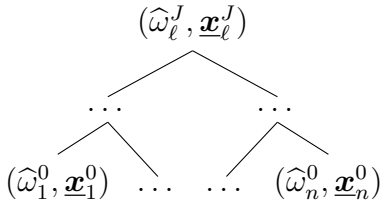
$$(\omega_k^j, \underline{\boldsymbol{x}}_k^j) \qquad \text{for} \quad 1 \leq k \leq \frac{n}{2^j}$$

satisfying

$$\omega_k^j := \left(\omega_{2k-1}^{j-1} + \omega_{2k}^{j-1}\right)$$

$$\underline{\boldsymbol{x}}_k^j := \frac{\left(\omega_{2k-1}^{j-1}\underline{\boldsymbol{x}}_{2k-1}^{j-1} + \omega_{2k}^{j-1}\underline{\boldsymbol{x}}_{2k}^{j-1}\right)}{\left(\omega_{2k-1}^{j-1} + \omega_{2k}^{j-1}\right)}$$

Then, given a root weight update $\omega_\ell^J \to \widehat{\omega}_\ell^J$*, we call* $\widehat{T}$ *a weight-updated tree, given by*



if $\widehat{\omega}_k^j = \theta_k^j \widehat{\omega}_\ell^J$*, where* $\theta_k^j = \omega_k^j / \omega_\ell^J$ *for each level* $j \in [\![0, J]\!]$*.*

3.1.0.8 Reduced Tree Measure

Definition 3.1.9. *Let $L, M, N \in \mathbb{N}$ be such that $N > L \geq M+1$. Let $\mu \in \mathcal{M}_+(\mathbb{R}^M)$, with $supp(\mu) \subseteq K$, $K \in \mathcal{B}(\mathbb{R}^M)$ compact, be given by*

$$\mu = \sum_{i=1}^{N} \lambda_i \delta_{\underline{\boldsymbol{z}}_i}, \qquad \sum_{i=1}^{N} \lambda_i = 1$$

and let $\eta_L \in \mathcal{M}_+(\mathbb{R}^M)$, with $supp(\eta_L) \subseteq Conv\big(\{\underline{\boldsymbol{z}}_1, \ldots, \underline{\boldsymbol{z}}_N\}\big) \subseteq K$, denote an L-tree form of μ, given by

$$\eta_L = \sum_{i=1}^{L} \beta_i \delta_{\underline{\boldsymbol{y}}_i}, \quad \beta_i = P(T_i),\ \sum_{i=1}^{L} \beta_i = 1,\ \underline{\boldsymbol{y}}_i = R(T_i)$$

where $\{T_1, \ldots, T_L\}$ denotes a forest of complete binary trees.

Then, for some $M^{'} \leq M$, we call $\eta_{M'} \in \mathcal{M}_+(\mathbb{R}^M)$ a reduced tree measure for η_L if

1. *$\eta_{M'}$ is a reduced measure for η_L, given by*

$$\eta_{M'} = \sum_{i=1}^{M^{'}} \widehat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}} \qquad \widehat{\beta}_k \geq 0,\ \sum_{k=1}^{M^{'}} \widehat{\beta}_k = 1$$

2. *$\widehat{\beta}_k = P(\widehat{T}_{i_k}),\ \ \underline{\boldsymbol{y}}_{i_k} = R(\widehat{T}_{i_k}) \quad$ for $1 \leq k \leq M^{'}$*

   where each $\widehat{T}_{i_k}$ is a weight-updated tree T_{i_k}, corresponding to a root weight update $\beta_k \to \widehat{\beta}_k$ and $\{T_{i_1}, \ldots, T_{i_{M'}}\} \subset \{T_1, \ldots, T_L\}$.

Lemma 3.1.10. *Let $L, M, N \in \mathbb{N}$ be such that $N > L \geq M+1$. Let $\mu \in \mathcal{M}_+(\mathbb{R}^M)$, with $supp(\mu) \subseteq K$, $K \in \mathcal{B}(\mathbb{R}^M)$ compact, be given by*

$$\mu = \sum_{i=1}^{N} \lambda_i \delta_{\underline{\boldsymbol{z}}_i}, \qquad \sum_{i=1}^{N} \lambda_i = 1$$

Given an L-tree form of μ , $\eta_L \in \mathcal{M}_+(\mathbb{R}^M)$ and given some $M^{'} \leq M$, a reduced tree measure $\eta_{M'} \in \mathcal{M}_+(\mathbb{R}^M)$ for η_L is given by

$$\eta_{M'} = \sum_{i=1}^{M^{'}} \widehat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}} \qquad \widehat{\beta}_k = P(\widehat{T}_{i_k}),\ \ \underline{\boldsymbol{y}}_{i_k} = R(\widehat{T}_{i_k})$$

where each tree $\widehat{T}_{i_k}$ *is a weight-updated tree* T_{i_k} *in the* $supp(\eta_L)$*, corresponding to the root weight update* $\beta_k \to \widehat{\beta}_k$*.*

Then, by letting $\ell(T_{i_k}) = \{\underline{\boldsymbol{z}}_{i_j}\}_j$ *denote the collection of leaf nodes of the tree* $\widehat{T}_{i_k}$ *, we have*

$$\widehat{\mu} := \sum_{k=1}^{N'} \widehat{\lambda}_k \delta_{\underline{\boldsymbol{z}}_{i_k}} \qquad \widehat{\lambda}_j \geq 0,\ \sum_{k=1}^{N'} \widehat{\lambda}_k = 1,\ \{\underline{\boldsymbol{z}}_{i_k}\}_{k=1}^{N'} = \cup_{k=1}^{M'} \ell(T_{i_k})$$

where $\widehat{\lambda}_k = \theta_j^0 \widehat{\beta}_k$ *and* θ_k^0 *is defined, as in* (3.1.8)*. Moreover,* $\widehat{\mu}$ *satisfies*

$$\begin{aligned} &1) \quad \mathcal{C}oM(\widehat{\mu}) = \mathcal{C}oM(\mu) \\ &2) \quad supp(\widehat{\mu}) \subset supp(\mu) \\ &3) \quad card(supp(\widehat{\mu})) < card(supp(\mu)) \end{aligned}$$

Proof. By definition of a reduced tree measure $\eta_{M'}$ for η_L we have

$$\eta_{M'} = \sum_{i=1}^{M'} \widehat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}} \qquad \widehat{\beta}_k = P(\widehat{T}_{i_k}),\ \ \underline{\boldsymbol{y}}_{i_k} = R(\widehat{T}_{i_k})$$

where each tree $\widehat{T}_{i_k}$ is a weight-updated tree T_{i_k}, corresponding to the root weight update $\beta_k \to \widehat{\beta}_k$. Let us assume, without loss of generality that

$$depth(\widehat{T}_{i_1}) = \ldots = depth(\widehat{T}_{i'_M}) = J \leq D$$

where $D = \lceil log_2\big(\frac{N}{L}\big) \rceil$. By definition (3.1.8) of a weight-updated tree, each level $j \in [\![0, J]\!]$ of the binary forest $\{\widehat{T}_{i_1}, \ldots, \widehat{T}_{i_{M'}}\}$ contains the node pairs

$$(\widehat{\omega}_k^j, \underline{\boldsymbol{x}}_k^j) \qquad \text{for} \quad 1 \leq j \leq M' \cdot 2^{J-j}$$

satisfying

$$\widehat{\omega}_k^j := \big(\widehat{\omega}_{2k-1}^{j-1} + \widehat{\omega}_{2k}^{j-1}\big), \qquad \widehat{\omega}_k^j \geq 0,\ \sum_{k=1}^{M' 2^{J-j}} \widehat{\omega}_k^j = 1$$

and since $\underline{\boldsymbol{y}}_{i_k} = \underline{\boldsymbol{x}}_k^J$, together with its children nodes $\underline{\boldsymbol{x}}_k^j$, are produced by an application of the RHFC Algorithm to μ, we have

$$\underline{\boldsymbol{x}}_k^j = \frac{(\omega_{2k-1}^{j-1} \underline{\boldsymbol{x}}_{2k-1}^{j-1} + \omega_{2k}^{j-1} \underline{\boldsymbol{x}}_{2k}^{j-1})}{(\omega_{2k-1}^{j-1} + \omega_{2k}^{j-1})}$$

where $(\omega_k^0, \underline{\boldsymbol{x}}_k^0) = (\lambda_k, \underline{\boldsymbol{z}}_k)$ for $1 \leq k \leq N$.

Then, by construction we have

$$\mathcal{C}oM(\eta_{M'}) = \sum_{i=1}^{M'} \widehat{\beta}_k \underline{\boldsymbol{y}}_{i_k} = \sum_{k=1}^{M'} \widehat{\omega}_k^J \underline{\boldsymbol{x}}_k^J$$

Moreover, it is readily verified that for each level $j \in [\![0, J]\!]$ of the binary forest $\{\widehat{T}_{i_1}, \ldots, \widehat{T}_{i_{M'}}\}$, the node pairs $(\widehat{\omega}_k^j, \underline{\boldsymbol{x}}_k^j)$ satisfy

$$\sum_{k=1}^{M' 2^{J-j}} \widehat{\omega}_k^j \underline{\boldsymbol{x}}_k^j = \mathcal{C}oM(\eta_{M'}) \tag{3.10}$$

Let us verify (3.10), for $j = J-1$. By definition of $\underline{\boldsymbol{x}}_k^J$, we have

$$\underline{\boldsymbol{x}}_k^J = \frac{\left(\omega_{2k-1}^{J-1} \underline{\boldsymbol{x}}_{2k-1}^{J-1} + \omega_{2k}^{J-1} \underline{\boldsymbol{x}}_{2k}^{J-1}\right)}{\left(\omega_{2k-1}^{J-1} + \omega_{2k}^{J-1}\right)}$$

$$= \frac{\left((\beta_k \theta_{2k-1}^{J-1}) \underline{\boldsymbol{x}}_{2k-1}^{J-1} + (\beta_k \theta_{2k}^{J-1}) \underline{\boldsymbol{x}}_{2k}^{J-1}\right)}{\beta_k}$$

$$= \theta_{2k-1}^{J-1} \underline{\boldsymbol{x}}_{2k-1}^{J-1} + \theta_{2k}^{J-1} \underline{\boldsymbol{x}}_{2k}^{J-1}$$

since $\omega_j^{J-1} = \beta_k \theta_j^{J-1}$ and $(\omega_{2k-1}^{J-1} + \omega_{2k}^{J-1}) = \omega_k^J := \beta_k$ and hence we have

$$\begin{aligned}
\mathcal{C}oM(\eta_{M'}) &= \sum_{k=1}^{M'} \widehat{\omega}_k^J \underline{\boldsymbol{x}}_k^J \\
&= \sum_{k=1}^{M'} (\widehat{\beta}_k \theta_{2k-1}^{J-1}) \underline{\boldsymbol{x}}_{2k-1}^{J-1} + \sum_{k=1}^{M'} (\widehat{\beta}_k \theta_{2k}^{J-1}) \underline{\boldsymbol{x}}_{2k}^{J-1} \\
&= \sum_{k=1}^{2M'} \widehat{\omega}_k^{J-1} \underline{\boldsymbol{x}}_k^{J-1}
\end{aligned}$$

since $\widehat{\omega}_k^J = \widehat{\beta}_k$ and $\widehat{\omega}_j^{J-1} = \widehat{\beta}_k \theta_j^{J-1}$

Similarly, (3.10) can be verified for all $j \in [\![0, D]\!]$, in particular we note that

$$\mathcal{C}oM(\eta_{M'}) = \sum_{k=1}^{M' 2^J} \widehat{\omega}_k^0 \underline{\boldsymbol{x}}_k^0 \qquad \widehat{\omega}_k^0 \geq 0, \ \sum_{k=1}^{M' 2^J} \widehat{\omega}_k^0 = 1$$

Moreover for $1 \leq k \leq M^{'}2^{J}$, by construction we have $\underline{\boldsymbol{x}}_k^0 = \underline{\boldsymbol{z}}_{i_k} \in supp(\mu)$, hence we define

$$\sum_{k=1}^{M^{'}2^{J}} \widehat{\omega}_k^0 \delta_{\underline{\boldsymbol{z}}_{i_k}} \qquad \widehat{\omega}_k^0 \geq 0, \ \sum_{k=1}^{M^{'}2^{J}} \widehat{\omega}_k^0 = 1$$

Then, by definition of the L-Tree form η_L and the reduced tree measure $\eta_{M'}$ for η_L, we have

$$\mathcal{C}oM(\mu) = \mathcal{C}oM(\eta_L) = \mathcal{C}oM(\eta_{M'})$$

and since

$$\mathcal{C}oM(\eta_{M'}) = \mathcal{C}oM(\widehat{\mu})$$

we can conclude that

$$\begin{aligned}
&1) \quad \mathcal{C}oM(\widehat{\mu}) = \sum_{k=1}^{M^{'}2^{J}} \widehat{\omega}_k^0 \underline{\boldsymbol{z}}_{i_k} = \sum_{i=1}^{N} \lambda_i \underline{\boldsymbol{z}}_i = \mathcal{C}oM(\mu) \\
&2) \quad supp(\widehat{\mu}) \subset supp(\mu) \\
&3) \quad card(supp(\widehat{\mu})) = M^{'}2^{J} < \frac{M^{'}N}{L} < N = card(supp(\mu))
\end{aligned}$$

□

3.1.0.9 Tree Splitting

Lemma 3.1.11. *Let $L, M, N \in \mathbb{N}$ be such that $N > L \geq M+1$. Let $\mu \in \mathcal{M}_+(\mathbb{R}^M)$, with $supp(\mu) \subseteq K$, $K \in \mathcal{B}(\mathbb{R}^M)$ compact, be given by*

$$\mu = \sum_{i=1}^{N} \lambda_i \delta_{\underline{\boldsymbol{z}}_i}, \qquad \sum_{i=1}^{N} \lambda_i = 1$$

and let $\eta_L \in \mathcal{M}_+(\mathbb{R}^M)$, with $supp(\eta_L) \subseteq Conv\big(\{\underline{\boldsymbol{z}}_1, \ldots, \underline{\boldsymbol{z}}_N\}\big) \subseteq K$, denote an L-tree form of μ, given by

$$\eta_L = \sum_{i=1}^{L} \beta_i \delta_{\underline{\boldsymbol{y}}_i}, \quad \beta_i = P(T_i), \ \sum_{i=1}^{L} \beta_i = 1, \ \underline{\boldsymbol{y}}_i = R(T_i)$$

where $\{T_1, \ldots, T_L\}$ denotes a forest of complete binary trees, of $depth(T_i) = J$ for $1 \leq i \leq s$ and $depth(T_i) = J-1$ for $s+1 \leq i \leq L$ with $1 \leq J \leq D$, $D = \lceil log_2\big(\frac{N}{L}\big) \rceil$

Then, there exists an L-tree form of μ, $\eta_L^{'} \in \mathcal{M}_+(\mathbb{R}^M)$

$$\eta_L^{'} = \sum_{i=1}^{L} \beta_i^{'} \delta_{\underline{\boldsymbol{y}}_i^{'}}, \quad \beta_i = P(\tilde{T}_i), \ \sum_{i=1}^{L} \beta_i^{'} = 1, \ \underline{\boldsymbol{y}}_i^{'} = R(\tilde{T}_i)$$

where $\{\tilde{T}_1, \ldots, \tilde{T}_L\}$ denotes a re-weighted sub-collection of sub-trees of $\{T_1, \ldots, T_L\}$, of $depth(\tilde{T}_i) = J-1$ for $1 \leq i \leq q$ and $depth(\tilde{T}_i) = J-2$ for $q+1 \leq i \leq L$.

In particular, $\tilde{T}_i = \widehat{T}_{i_k}$ or $\tilde{T}_i = \widehat{T}_{i_k}^{\ell}$ or $\tilde{T}_i = \widehat{T}_{i_k}^{r}$, where $\widehat{T}_{i_k}$ is a weight-updated tree $T_{i_k} \in \{T_1, \ldots, T_L\}$ and $\{\widehat{T}_{i_k}^{\ell}, \widehat{T}_{i_k}^{r}\}$ denote the left and right sub-trees of $\widehat{T}_{i_k}$, of depth $depth(\widehat{T}_{i_k}) - 1$.

Proof. Assume that we have an L-Tree form of μ, given by

$$\eta_L = \sum_{i=1}^{L} \beta_i \delta_{\underline{\boldsymbol{y}}_i}, \quad \beta_i = P(T_i), \ \sum_{i=1}^{L} \beta_i = 1, \ \underline{\boldsymbol{y}}_i = R(T_i)$$

where $\{T_1, \ldots, T_L\}$ denotes a forest of complete binary trees, of $depth(T_i) = J$ for $1 \leq i \leq s$ and $depth(T_i) = J-1$ for $s+1 \leq i \leq L$ with $1 \leq J \leq D$, $D = \lceil log_2\big(\frac{N}{L}\big) \rceil$

Then, by lemma (3.0.28), there exists a reduced tree measure $\eta_{M'}$ for η_L, for some $M^{'} \leq M$

$$\eta_{M'} = \sum_{i=1}^{M^{'}} \widehat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}} \qquad \widehat{\beta}_k = P(\widehat{T}_{i_k}), \ \sum_{k=1}^{M^{'}} \widehat{\beta}_k = 1, \ \underline{\boldsymbol{y}}_{i_k} = R(\widehat{T}_{i_k})$$

where each tree $\widehat{T}_{i_k}$ is a weight-updated tree T_{i_k}, corresponding to the root weight update $\beta_k \rightarrow \widehat{\beta}_k$.

By the definition of $\widehat{T}_{i_k}$, we know that if $B = \mathit{depth}(\widehat{T}_{i_k})$, then we have $\widehat{\beta}_k = \widehat{\omega}_k^B$ and $\underline{\boldsymbol{y}}_{i_k} = \underline{\boldsymbol{x}}_k^B$

$$P(\widehat{T}_{i_k})R(\widehat{T}_{i_k}) = \widehat{\omega}_k^B \underline{\boldsymbol{x}}_k^B = \widehat{\omega}_k^B \left(\frac{\omega_{2k-1}^{B-1} \underline{\boldsymbol{x}}_{2k-1}^{B-1} + \omega_{2k}^{B-1} \underline{\boldsymbol{x}}_{2k}^{B-1}}{\omega_{2k-1}^{B-1} + \omega_{2k}^{B-1}} \right) \tag{3.11}$$

$$= \widehat{\omega}_k^B \left(\frac{(\omega_k^B \theta_{2k-1}^{B-1}) \underline{\boldsymbol{x}}_{2k-1}^{B-1} + (\omega_k^B \theta_{2k}^{B-1}) \underline{\boldsymbol{x}}_{2k}^{B-1}}{\omega_k^B} \right)$$

$$= (\widehat{\omega}_k^B \theta_{2k-1}^{B-1}) \underline{\boldsymbol{x}}_{2k-1}^{B-1} + (\widehat{\omega}_k^r \theta_{2k}^{B-1}) \underline{\boldsymbol{x}}_{2k}^{B-1}$$

$$= \widehat{\omega}_{2k-1}^{B-1} \underline{\boldsymbol{x}}_{2k-1}^{B-1} + \widehat{\omega}_{2k}^{B-1} \underline{\boldsymbol{x}}_{2k}^{B-1}$$

$$= P(\widehat{T}_{i_k}^{\ell})R(\widehat{T}_{i_k}^{\ell}) + P(\widehat{T}_{i_k}^{r})R(\widehat{T}_{i_k}^{r})$$

since $\omega_j^{B-1} = \omega_k^B \theta_j^{B-1}$ and $\widehat{\omega}_j^{B-1} = \widehat{\omega}_k^B \theta_j^{B-1}$, where $\widehat{T}_{i_k}^{\ell}$ and $\widehat{T}_{i_k}^{r}$ are the two constituent left and right sub-trees of T_{i_k} of depth $(B-1)$.

We proceed by selecting $(L - M') \leq M'$ trees of greatest depth $\{\widehat{T}_{i_{j_1}}, \ldots, \widehat{T}_{i_{j_{L-M'}}}\} \subseteq \{\widehat{T}_{i_1}, \ldots, \widehat{T}_{i_{M'}}\}$, from the support of $\eta_{M'}$. Subsequently, for each tree selected we remove the root pair $\left(P(\widehat{T}_{i_{j_k}}),\ R(\widehat{T}_{i_{j_k}})\right)$ and whence split the tree $\widehat{T}_{i_{j_k}}$ into the two constituent sub-trees

$$\left(P(\widehat{T}_{i_{j_k}}),\ R(\widehat{T}_{i_{j_k}})\right)$$

$$\left(P(\widehat{T}_{i_{j_k}}^{\ell}),\ R(\widehat{T}_{i_{j_k}}^{\ell})\right) \qquad \left(P(\widehat{T}_{i_{j_k}}^{r}),\ R(\widehat{T}_{i_{j_k}}^{r})\right)$$

Next, we set

$$\{\tilde{T}_1, \ldots, \tilde{T}_L\} = \left(\{\widehat{T}_{i_1}, \ldots, \widehat{T}_{i_{M'}}\} \backslash \{\widehat{T}_{i_{j_1}}, \ldots, \widehat{T}_{i_{j_{L-M'}}}\}\right) \cup \{\widehat{T}_{i_{j_k}}^{\ell}, \widehat{T}_{i_{j_k}}^{r}, \ldots, \widehat{T}_{i_{j_{L-M'}}}^{\ell}, \widehat{T}_{i_{j_{L-M'}}}^{r}\}$$

and thereby obtain $\eta_L^{'}$, given by

$$\eta_L^{'} = \sum_{i=1}^{L} \beta_i^{'} \delta_{\underline{\boldsymbol{y}}_i^{'}}, \quad \beta_i^{'} = P(\tilde{T}_i),\ \sum_{i=1}^{L} \beta_i^{'} = 1, \quad \underline{\boldsymbol{y}}_i^{'} = R(\tilde{T}_i)$$

where $depth(\tilde{T}_i) = J-1$ or $J-2$

By construction of $\eta_L^{'}$, we have $\tilde{T}_i = \widehat{T}_{i_k}$ or $\tilde{T}_i = \widehat{T}_{i_k}^{\ell}$ or $\tilde{T}_i = \widehat{T}_{i_k}^{r}$. In any of the aforementioned cases, by definition of a weight-updated tree, the root node is given by

$$\underline{\boldsymbol{y}}_{i_k} = R(\tilde{T}_i) = \underline{\boldsymbol{x}}_k^B$$

where $B = depth(\tilde{T}_i)$ and $\underline{\boldsymbol{x}}_k^B$, is produced by an application of the RHFC to μ and the associated root weight

$$\widehat{\beta}_k = P(\tilde{T}_i) = \widehat{\omega}_k^B$$

Moreover, by definition of a weight-updated tree for each level $j \in [\![0, J]\!]$, we have

$$\widehat{\omega}_r^j = \theta_r^j \widehat{\omega}_k^B = \frac{\omega_r^j}{\omega_k^B}\widehat{\omega}_k^B = \Big(\frac{\omega_{2r-1}^{j-1}}{\omega_k^B} + \frac{\omega_{2r}^{j-1}}{\omega_k^B}\Big)\widehat{\omega}_k^B = \theta_{2r-1}^{j-1}\widehat{\omega}_k^B + \theta_{2r}^{j-1}\widehat{\omega}_k^B = \widehat{\omega}_{2r-1}^{j-1} + \widehat{\omega}_{2r}^{j-1}$$

and since

$$\underline{\boldsymbol{x}}_r^j = \frac{\big(\omega_{2r-1}^{j-1}\underline{\boldsymbol{x}}_{2r-1}^{j-1} + \omega_{2r}^{j-1}\underline{\boldsymbol{x}}_{2r}^{j-1}\big)}{\big(\omega_{2r-1}^{j-1} + \omega_{2r}^{j-1}\big)} = \frac{\Big((\omega_k^B\theta_{2r-1}^{j-1})\underline{\boldsymbol{x}}_{2r-1}^{j-1} + (\omega_k^B\theta_{2r}^{j-1})\underline{\boldsymbol{x}}_{2r}^{j-1}\Big)}{\big(\omega_k^B\theta_{2r-1}^{j-1} + \omega_k^B\theta_{2r}^{j-1}\big)}$$

$$= \frac{\Big(\theta_{2r-1}^{j-1}\underline{\boldsymbol{x}}_{2r-1}^{j-1} + \theta_{2r}^{j-1}\underline{\boldsymbol{x}}_{2r}^{j-1}\Big)}{\big(\theta_{2r-1}^{j-1} + \theta_{2r}^{j-1}\big)} = \frac{\Big((\widehat{\omega}_k^B\theta_{2r-1}^{j-1})\underline{\boldsymbol{x}}_{2r-1}^{j-1} + (\widehat{\omega}_k^B\theta_{2r}^{j-1})\underline{\boldsymbol{x}}_{2r}^{j-1}\Big)}{\big(\widehat{\omega}_k^B\theta_{2r-1}^{j-1} + \widehat{\omega}_k^B\theta_{2r}^{j-1}\big)}$$

$$= \frac{\Big(\widehat{\omega}_{2r-1}^{j-1}\underline{\boldsymbol{x}}_{2r-1}^{j-1} + \widehat{\omega}_{2r}^{j-1}\underline{\boldsymbol{x}}_{2r}^{j-1}\Big)}{\big(\widehat{\omega}_{2r-1}^{j-1} + \widehat{\omega}_{2r}^{j-1}\big)}$$

By definition of a reduced tree measure $\eta_{M'}$, we have

$$\mathcal{C}oM(\eta_{M'}) = \mathcal{C}oM(\eta_L) = \mathcal{C}oM(\mu)$$

and by (3.11), we obtain

$$\mathcal{C}oM(\eta_L^{'}) = \mathcal{C}oM(\eta_{M'}) = \mathcal{C}oM(\mu)$$

and whence $\eta_L^{'}$ satisfies the definition of an L-Tree form of μ.

□

Let $L, M, N \in \mathbb{N}$ be such that $N > L \geq M+1$ and let $\mu := (\Psi_*\nu)$ and let the L-Tree form of μ, $\eta_L \in \mathcal{M}_+(\mathbb{R}^M)$, constructed via RHFC, be given and set $\eta_L^0 := \eta_L$. Let the reduced tree measure $\eta_{M'} \in \mathcal{M}_+(\mathbb{R}^M)$ for η_L, be given and set $\eta_{M'}^0 := \eta_{M'}$.

Then, applying lemma (3.1.10), yields a probability measure $\widehat{\mu} \in \mathcal{M}_+(\mathbb{R}^M)$ and we set $\widehat{\mu}^1 := \widehat{\mu}$, given by

$$\widehat{\mu}^1 = \sum_{k=1}^{N_1} \widehat{\omega}_k^1 \delta_{\Psi(\underline{\boldsymbol{x}}_{i_k})} \qquad \widehat{\omega}_k^1 \geq 0,\ \sum_{k=1}^{N_1} \widehat{\omega}_k^1 = 1$$

By construction, $\widehat{\mu}^1$ satisfies

$$1) \quad \mathcal{C}oM(\widehat{\mu}^1) = \sum_{k=1}^{N_1} \widehat{\omega}_k^1 \Psi(\underline{\boldsymbol{x}}_{i_k}) = \sum_{i=1}^{N} \omega_i \Psi(\underline{\boldsymbol{x}}_i) = \mathcal{C}oM(\mu)$$

$$2) \quad \{\Psi(\underline{\boldsymbol{x}}_{i_k})\}_{k=1}^{N_1} \subset \{\Psi(\underline{\boldsymbol{x}}_i)\}_{i=1}^{N}$$

$$3) \quad \mathit{card(supp}(\widehat{\mu}^1)) = N_1 < N = \mathit{card(supp}(\mu))$$

where $N_1 \approx M^{'}2^D$, $D = \lceil log_2\big(\frac{N}{L}\big) \rceil$

Then, we proceed by recursion, for $j \geq 0$, letting $\eta_{M'}^j \in \mathcal{M}_+(\mathbb{R}^M)$ denote the reduced tree measure for η_L^j and utilising lemma (3.1.11) to construct η_L^{j+1}. We note that at each step of the aforementioned recursion, the construction of $\eta_{M'}^j$ from η_L^j eliminates $(L - M^{'})$ trees of approximate depth $(D - j)$ from $\mathit{supp}(\eta_L^j)$ and the construction of η_L^{j+1} trims the root nodes of the collection of trees in the $\mathit{supp}(\eta_{M'}^j)$, thereby reducing the depth of the underlying binary forest at every iteration. We terminate the recursion, when $\eta_{M'}^j$ is supported a forest of binary trees of depth 0, i.e. when we have reached the leaf node level.

We observe that with each construction of $\eta_{M'}^j$, as a result of lemma (3.1.10), a new probability measure $\widehat{\mu}^j \in \mathcal{M}_+(\mathbb{R}^M)$, is generated by the leaf pairs of $\eta_{M'}^j$. By con-

struction $\widehat{\mu}^j$ satisfies

$$1) \quad \mathcal{C}oM(\widehat{\mu}^j) = \sum_{k=1}^{N_j} \widehat{\omega}_k^j \Psi(\underline{\boldsymbol{x}}_{i_k}) = \sum_{i=1}^{N} \omega_i \Psi(\underline{\boldsymbol{x}}_i) = \mathcal{C}oM(\mu)$$

$$2) \quad \{\Psi(\underline{\boldsymbol{x}}_{i_k})\}_{k=1}^{N_j} \subset \{\Psi(\underline{\boldsymbol{x}}_{i_k})\}_{k=1}^{N_{j-1}} \subset \{\Psi(\underline{\boldsymbol{x}}_i)\}_{i=1}^{N}$$

$$3) \quad card(supp(\widehat{\mu}^j)) = N_j < N_{j-1} < N = card(supp(\mu))$$

By lemma (3.1.10), $\widehat{\mu}^j$ arises from the union of leaf nodes of $\eta_{M'}^j$, supported on the binary forest $\{\widehat{T}_{i_1}^j, \ldots, \widehat{T}_{i_{M'}}^j\}$ of approximate depth $(D-j)$ and whence $card(supp(\widehat{\mu}^j)) = N_j \approx M' 2^{(D-j)}$. Moreover, by definition the union of leaf nodes of $\eta_{M'}^j$ is a sub-collection of the union of leaf nodes of $\eta_{M'}^{j-1}$ and hence (2) follows, for each $j \geq 0$. And since by construction, we have

$$\mathcal{C}oM(\eta_L^j) = \mathcal{C}oM(\eta_{M'}^j) = \mathcal{C}oM(\mu)$$

and

$$\mathcal{C}oM(\eta_{M'}^j) = \mathcal{C}oM(\widehat{\mu}^j)$$

therefore (1) follows.

We note that each measure $\widehat{\mu}^j$ gives rise to a measure

$$\widehat{\nu}^j = \sum_{k=1}^{N_j} \widehat{\omega}_k^j \delta_{\underline{\boldsymbol{x}}_{i_k}} \qquad \widehat{\omega}_k^j \geq 0, \ \sum_{k=1}^{N_j} \widehat{\omega}_k^j = 1$$

and by conditions 1) and 2) of $\widehat{\mu}^j$ it follows that $\widehat{\nu}^j$ is Ψ-generalised cubature measure for ν. Whence, the aforementioned recursion yields a sequence $\{\widehat{\nu}^j\}_j$ of Ψ-generalised cubature measures for ν of decreasing cardinalities. We terminate, when $\eta_{M'}^j$ is supported on a binary forest of depth 0, that is, it is supported on a sub-collection of nodes in μ, yielding $card(supp(\widehat{\mu}^j)) = M' \leq M + 1$ and hence we have $card(supp(\widehat{\nu}^j)) = M'$. Therefore, since the last $\widehat{\nu}^j$ satisfies condition 3) in (3.19), we obtain a Ψ-generalised Carathéodory cubature measure for ν, $\widehat{\nu}_\Psi$.

3.1.0.10 Construction of Reduced Tree Measures

The Recombination Algorithm, first introduced in [96], combines all the aforementioned ideas in this chapter and whence can be exploited to construct a Ψ-generalised Carathéodory cubature measure for $\widehat{\nu}_{\Psi}$ for an atomic probability measure ν and a continuous, ν-measurable map $\Psi = (\psi_1, \ldots, \psi_M)$. However, prior to introducing the Recombination Algorithm we require a method for generating reduced tree measures $\eta_{M'}$ for η_L.

In the following section we will introduce two constructive algorithms for generating reduced tree measures $\eta_{M'}$ for an η_L, as defined in definition (3.1.9), for $L = M+1$ and $L = 2M$. The method for generating a reduced tree measure $\eta_{M'}$ for η_L, consists of two phases:

1. Construct a reduced measure $\tilde{\eta}_{M'}$ for η_L, as in definition (3.0.27)
2. Update the weights of every tree in the support of $\tilde{\eta}_{M'}$, by utilising the definition (3.1.8) and hence produce the reduced tree measure $\eta_{M'}$.

To perform the first phase, we introduce the 1-Tree Measure Reduction Algorithm and M-Tree Measure Reduction Algorithm, respectively, for $L = M + 1$ and $L = 2M$.

Letting η_L be given by

$$\eta_L = \sum_{i=1}^{L} \beta_i \delta_{\underline{\boldsymbol{y}}_i}, \quad \beta_i = P(T_i) \geq 0, \ \sum_{i=1}^{L} \beta_i = 1, \ \underline{\boldsymbol{y}}_i = R(T_i)$$

and setting

$$\begin{aligned} A &:= [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_{M+1}] \in \mathbb{R}^{M \times L} \\ \underline{\boldsymbol{\beta}} &:= [\beta_1, \beta_2, \ldots, \beta_{M+1}]^T \in \mathbb{R}_+^L \\ \underline{\boldsymbol{b}} &:= \mathcal{C}oM(\eta_L) \in \mathbb{R}^M \end{aligned}$$

Both of the aforementioned measure reduction algorithms, provide an alternative method to the Simplex Algorithm, for finding a basic feasible solution, with $M^{'} \leq M$, $\widehat{\underline{\boldsymbol{\beta}}} = [\beta_1, \ldots, \beta_{M'}, 0, \ldots, 0]^T \in \mathbb{R}_+^L$ to the LP problem

$$A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}$$

$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}$$

and thereby generating a reduced measure $\tilde{\eta}_{M'}$, given by

$$\tilde{\eta}_{M'} = \sum_{i=1}^{M'} \widehat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}}, \quad \widehat{\beta}_k \geq 0, \ \sum_{i=1}^{M'} \widehat{\beta}_k = 1, \ \underline{\boldsymbol{y}}_{i_k} = R(T_{i_k})$$

then we proceed to updating the weights down each node of every tree $T_{i_k} \in \{T_{i_1}, \ldots, T_{i_{M'}}\}$ in the $supp(\tilde{\eta}_{M'})$. And whence, we produce the binary forest $\{\widehat{T}_{i_1}, \ldots, \widehat{T}_{i_{M'}}\}$, where each tree $\widehat{T}_{i_k}$ is a weight-updated tree T_{i_k}, corresponding to the root weight update $\beta_k \to \widehat{\beta}_k$, as defined in (3.1.8), yielding the reduced tree measure

$$\eta_M = \sum_{i=1}^{M} \widehat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}} \qquad \widehat{\beta}_k = P(\widehat{T}_{i_k}) \geq 0, \ \sum_{i=1}^{M'} \widehat{\beta}_k = 1, \ \underline{\boldsymbol{y}}_{i_k} = R(\widehat{T}_{i_k})$$

The 1-Tree Measure Reduction Algorithm, integrated into the Recombination Algorithm, was first introduced in [96]. This algorithm enables us to construct the Ψ-generalised Carathéodory cubature measure for ν, by sequentially clustering the original measure ν into $(M+1)$ binary trees at every iteration.

The novelty of the M-Tree Measure Reduction, is that, whilst it has the same theoretical computational complexity as the 1-Tree Measure Reduction, when integrated into the Recombination Algorithm, the overall theoretical computational complexity of the Recombination Algorithm is reduced by one order of magnitude. The key, behind this novel construction is to cluster the original measure ν into $2M$ binary trees at every iteration, which will result in halving the support of the original measure ν with every iteration.

Moreover, from empirical results in the Chapter 5, the Recombination Algorithm implemented with M-Tree Measure Reduction, turns out to be one order of magnitude faster than the Recombination Algorithm with the Dual Simplex or Interior Point Methods.

3.2 1-Tree Measure Reduction Algorithm

Algorithm 4: 1-Tree Measure Reduction

Input: Let η_{M+1} be an atomic probability measure on $\mathbb{R}^M$, given by

$$\eta_{M+1} = \sum_{i=1}^{M+1} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$$

1) Set

$$A^{(0)} := [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_{M+1}]$$

$$\underline{\boldsymbol{\beta}}^{(0)} := [\beta_1, \beta_2, \ldots, \beta_{M+1}]^T$$

2) Compute $M^{'} = rank(A^{(0)})$

For $i = 1 : (M+1) - M^{'}$

3) Produce a null vector $\underline{\boldsymbol{\phi}} \in \mathcal{K}er(A^{(i-1)})$

4) Compute $\alpha_{(i)}$

$$\alpha_{(i)} := \min_{1 \le j \le (M+1)-(i-1)} \left\{ \frac{\beta_j^{(i-1)}}{\phi_j} : \ \phi_j > 0 \right\} = \frac{\beta_{k^{(i)}}^{(i-1)}}{\phi_{k^{(i)}}}$$
$$\text{for some } k^{(i)} \in [\![1, (M+1) - (i-1)]\!]$$

5) Set $\underline{\boldsymbol{\beta}}^{(i)} := [\beta_1^{(i)}, \beta_2^{(i)}, \ldots, \beta_{(M+1)-i}^{(i)}]^T$

$$\beta_j^{(i)} := \begin{cases} \beta_j^{(i-1)} - \alpha_{(i)}\phi_j & \text{for } j \in [\![1, k^{(i)} - 1]\!] \\ \beta_{j+1}^{(i-1)} - \alpha_{(i)}\phi_{j+1} & \text{for } j \in [\![k^{(i)}, (M+1) - i]\!] \end{cases}$$

6) Set $A^{(i)} := [\underline{\boldsymbol{y}}_1^{(i)}, \underline{\boldsymbol{y}}_2^{(i)}, \ldots, \underline{\boldsymbol{y}}_{(M+1)-i}^{(i)}]$

$$\underline{\boldsymbol{y}}_j^{(i)} := \begin{cases} \underline{\boldsymbol{y}}_j^{(i-1)} & \text{for } j \in [\![1, k^{(i)} - 1]\!] \\ \underline{\boldsymbol{y}}_{j+1}^{(i-1)} & \text{for } j \in [\![k^{(i)}, (M+1) - i]\!] \end{cases}$$

end i **loop**

7) Set

$$\underline{\hat{\boldsymbol{\beta}}} = [\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_{M'}]^T := [\beta_1^{(M+1)-M'}, \beta_2^{(M+1)-M'}, \ldots, \beta_{M'}^{(M+1)-M'}]^T = \underline{\boldsymbol{\beta}}^{(M+1)-M'}$$

8) Set

$$\hat{A} = [\underline{\boldsymbol{y}}_{i_1}, \underline{\boldsymbol{y}}_{i_2}, \ldots, \underline{\boldsymbol{y}}_{i_{M'}}] := [\underline{\boldsymbol{y}}_1^{(M+1)-M'}, \underline{\boldsymbol{y}}_2^{(M+1)-M'}, \ldots, \underline{\boldsymbol{y}}_{M'}^{(M+1)-M'}] = A^{(M+1)-M'}$$

Output: Reduced measure for η_{M+1}

$$\tilde{\eta}_{M'} = \sum_{k=1}^{M'} \hat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}}$$

The following provides a constructive proof of lemma (3.0.28) for $L = M + 1$.

Proof. Let $\eta_{M+1} = \sum_{i=1}^{M+1} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$ an atomic probability measure on $\mathbb{R}^M$. Set

$$\begin{aligned} &\underline{\boldsymbol{y}}_j = [1, y_{2,j} \ldots, y_{M,j}]^T \in \mathbb{R}^M \\ &A^{(0)} := [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_{M+1}] \in \mathbb{R}^{M \times (M+1)} \\ &\underline{\boldsymbol{\beta}}^{(0)} := [\beta_1, \beta_2, \ldots, \beta_{M+1}]^T \in \mathbb{R}_+^{(M+1)}, \ \sum_{j=1}^{M+1} \beta_j = 1 \end{aligned}$$

For $i = 1$, we proceed with step 2) of the 1-Tree Measure Reduction Algorithm. Compute rank$(A^{(0)}) = M' \leq M$ and hence $dim\big(\mathcal{K}er(A^{(0)})\big) = (M+1) - M'$. By step 3) we compute $\underline{\boldsymbol{\phi}} \in \mathcal{K}er(A^{(0)})$. Clearly, since $A^{(0)}\underline{\boldsymbol{\phi}} = 0$, we have

$$\sum_{j=1}^{M+1} \phi_j \underline{\boldsymbol{y}}_j = \underline{\mathbf{0}}$$

and in particular since $y_{1,j} = 1$, $\forall j \in [\![1, (M+1)]\!]$, we have

$$\sum_{j=1}^{M+1} \underline{\boldsymbol{\phi}}_j = 0$$

and hence at least one $\phi_j > 0$, as not all $\phi_j = 0$

$$\mathcal{C}oM(\eta_{M+1}) = \sum_{j=1}^{M+1} \beta_j \underline{\boldsymbol{y}}_j = \sum_{j=1}^{M+1} \beta_j \underline{\boldsymbol{y}}_j - \alpha \left(\sum_{j=1}^{M+1} \phi_j \underline{\boldsymbol{y}}_j \right) = \sum_{j=1}^{M+1} \big(\beta_j - \alpha\phi_j\big) \underline{\boldsymbol{y}}_j$$

holds for any $\alpha \in \mathbb{R}$, in particular for $i = 1$ it holds for $\alpha = \alpha_{(1)}$, defined in step 4), then we have $\beta_j - \alpha_{(1)}\phi_j \geq 0$ for $\forall j \in [\![1, (M+1)]\!]$, $\beta_{k^{(1)}} - \alpha_{(1)}\phi_{k^{(1)}} = 0$ for some $k^{(1)} \in [\![1, (M+1)]\!]$ and $\sum_{j=1}^{M+1} \big(\beta_j - \alpha_{(1)}\phi_j\big) = 1$

Hence, we have produced a probability vector $\big[\underline{\boldsymbol{\beta}}^{(0)} - \alpha_{(1)}\underline{\boldsymbol{\phi}}\big]$ placing zero mass on the $k^{(1)}th$ node in $supp(\eta_{M+1})$. Subsequently, we shrink the probability vector produced by removing the zeroed coordinate to obtain

$$\underline{\boldsymbol{\beta}}^{(1)} := [\beta_1^{(1)}, \beta_2^{(1)}, \ldots, \beta_M^{(1)}]^T \in \mathbb{R}_+^M, \ \sum_{j=1}^{M} \beta_j^{(1)} = 1$$

as defined in step 5). And since the $k^{(1)}th$ node $\underline{\boldsymbol{y}}_{k^{(1)}}$ can be represented as a linear combination of $\big\{\underline{\boldsymbol{y}}_1, \ldots, \underline{\boldsymbol{y}}_{k^{(1)}-1}, \underline{\boldsymbol{y}}_{k^{(1)}+1}, \cdots \underline{\boldsymbol{y}}_{M+1}\big\}$

$$\underline{\boldsymbol{y}}_{k^{(1)}} = -\frac{1}{\phi_{k^{(1)}}} \sum_{\substack{j=1 \\ j \neq k^{(1)}}}^{M+1} \phi_j \underline{\boldsymbol{y}}_j$$

we can eliminate the node $\underline{\boldsymbol{y}}_{k^{(1)}}$ from $A^{(0)}$, without altering the rank of the new coefficient matrix. We set $A^{(1)} = [\underline{\boldsymbol{y}}_1^{(1)}, \underline{\boldsymbol{y}}_2^{(1)}, \ldots, \underline{\boldsymbol{y}}_M^{(1)}] \in \mathbb{R}^{M \times M}$ as defined in step 6), with rank($A^{(0)}$)=rank($A^{(1)}$)=M'. Hence, we produce a new probability measure $\tilde{\eta}_M$, preserving the $\mathcal{C}oM(\eta_{M+1})$.

$$\tilde{\eta}_M = \sum_{j=1}^{M} \beta_j^{(1)} \delta_{\underline{\boldsymbol{y}}_j^{(1)}} : \ \big\{\underline{\boldsymbol{y}}_j^{(1)}\big\}_{j=1}^{M} \subset supp(\eta_{M+1}) \ , \ \mathcal{C}oM(\eta_{M+1}) = \mathcal{C}oM(\tilde{\eta}_M)$$

At the end of iteration $(i-1) \leq M - M^{'}$, we have produced

$$A^{(i-1)} := [\underline{\boldsymbol{y}}_1^{(i-1)}, \underline{\boldsymbol{y}}_2^{(i-1)}, \ldots, \underline{\boldsymbol{y}}_{(M+1)-(i-1)}^{(i-1)}] \in \mathbb{R}^{M \times ((M+1)-(i-1))},\ rank(A^{(i-1)}) = M^{'}$$

$$\underline{\boldsymbol{\beta}}^{(i-1)} := [\beta_1^{(i-1)}, \beta_2^{(i-1)}, \ldots, \beta_{(M+1)-(i-1)}^{(i-1)}]^T \in \mathbb{R}^{(M+1)-(i-1)},\ \sum_{j=1}^{(M+1)-(i-1)} \beta_j^{(i-1)} = 1$$

$$\tilde{\eta}_{(M+1)-(i-1)} = \sum_{j=1}^{(M+1)-(i-1)} \beta_j^{(i-1)} \delta_{\underline{\boldsymbol{y}}_j^{(i-1)}} :\ \{\underline{\boldsymbol{y}}_j^{(i-1)}\}_{j=1}^{(M+1)-(i-1)} \subset supp(\eta_{M+1}),$$

$$\mathcal{C}oM(\eta_{M+1}) = \mathcal{C}oM(\tilde{\eta}_{(M+1)-(i-1)})$$

Moreover, the dimension kernel of $A^{(i-1)}$ has the following bound

$$dim\big(\mathcal{K}er(A^{(i-1)})\big) = (M+1) - (i-1) - rank(A^{(i-1)}) = (M+1) - (i-1) - M^{'} \geq 1$$

Hence, for iteration i at step 3) we can compute $\underline{\boldsymbol{\phi}} \in \mathcal{K}er(A^{(i-1)})$ and since at least one $\phi_j > 0$, we have

$$\begin{aligned}
&\mathcal{C}oM(\tilde{\eta}_{(M+1)-(i-1)}) = \sum_{j=1}^{(M+1)-(i-1)} \beta_j^{(i-1)} \underline{\boldsymbol{y}}_j^{(i-1)} = \sum_{j=1}^{(M+1)-(i-1)} \beta_j^{(i-1)} \underline{\boldsymbol{y}}_j^{(i-1)} \\
&- \alpha_{(i)} \left(\sum_{j=1}^{(M+1)-(i-1)} \phi_j \underline{\boldsymbol{y}}_j^{(i-1)} \right) = \sum_{j=1}^{(M+1)-(i-1)} \big(\beta_j^{(i-1)} - \alpha_{(i)} \phi_j\big) \underline{\boldsymbol{y}}_j^{(i-1)} \\
&= \sum_{j=1}^{(M+1)-i} \beta_j^{(i)} \underline{\boldsymbol{y}}_j^{(i)} = \mathcal{C}oM(\tilde{\eta}_{(M+1)-i})
\end{aligned}$$

where $\alpha_{(i)}$ is as defined in step 4), $\beta_j^{(i)}$ is as defined in step 5) and $\underline{\boldsymbol{y}}_j^{(i)}$ is as defined in step 6). And since $\underline{\boldsymbol{y}}_{k^{(i)}}^{(i-1)}$ can be represented as a linear combination of $\{\underline{\boldsymbol{y}}_1^{(i-1)}, \ldots, \underline{\boldsymbol{y}}_{k^{(i)}-1}^{(i-1)}, \underline{\boldsymbol{y}}_{k^{(i)}+1}^{(i-1)}, \ldots \underline{\boldsymbol{y}}_{(M+1)-(i-1)}^{(i-1)}\}$

$$\underline{\boldsymbol{y}}_{k^{(i)}}^{(i-1)} = -\frac{1}{\phi_{k^{(i)}}} \sum_{\substack{j=1 \\ j \neq k^{(i)}}}^{(M+1)-(i-1)} \phi_j \underline{\boldsymbol{y}}_j^{(i-1)}$$

we can eliminate the node $\underline{\boldsymbol{y}}_{k^{(i)}}^{(i-1)}$ from $A^{(i-1)}$, without altering the rank of the new

coefficient matrix. Hence after iteration i, we have produced

$$A^{(i)} := [\underline{\boldsymbol{y}}_1^{(i)}, \underline{\boldsymbol{y}}_2^{(i)}, \ldots, \underline{\boldsymbol{y}}_{(M+1)-i}^{(i)}] \in \mathbb{R}^{M\times((M+1)-i)},\ rank(A^{(i)}) = M^{'}$$

$$\underline{\boldsymbol{\beta}}^{(i)} := [\beta_1^{(i)}, \beta_2^{(i)}, \ldots, \beta_{(M+1)-i}^{(i)}]^T \in \mathbb{R}^{(M+1)-i},\ \sum_{j=1}^{(M+1)-i} \beta_j^i = 1$$

$$\tilde{\eta}_{(M+1)-i} = \sum_{j=1}^{(M+1)-i} \beta_j^{(i)} \delta_{\underline{\boldsymbol{y}}_j^{(i)}} :\ \{\underline{\boldsymbol{y}}_j^{(i)}\}_{j=1}^{(M+1)-i} \subset supp(\tilde{\eta}_{M+1}),$$

$$\mathcal{C}oM(\tilde{\eta}_{M+1}) = \mathcal{C}oM(\tilde{\eta}_{(M+1)-i})$$

and whence at the end of iteration $i = (M+1) - M^{'}$, we have produced the reduced measure $\eta_{M'}$ for η_{M+1}

$$\tilde{\eta}_{M'} = \sum_{k=1}^{M^{'}} \beta_k^{(M+1-M^{'})} \delta_{\underline{\boldsymbol{y}}_k^{(M+1-M^{'})}} = \sum_{k=1}^{M^{'}} \hat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}} :\ \{\underline{\boldsymbol{y}}_{i_k}\}_{k=1}^{M^{'}} \subset supp(\eta_{M+1})\ ,$$

$$\sum_{k=1}^{M^{'}} \hat{\beta}_k = 1,\ \mathcal{C}oM(\eta_{M+1}) = \mathcal{C}oM(\tilde{\eta}_{M'})$$

□

Remark 3.2.1. *Theoretical Complexity of* 1*-Tree Measure Reduction Algorithm*

Let $C(\eta_{M+1}, \tilde{\eta}_{M'})$ denote the complexity to perform the 1*-Tree Measure Reduction Algorithm. Then $C(\eta_{M+1}, \tilde{\eta}_{M'})$ is dominated by the computation of the $\mathcal{K}er(A^{(i-1)})$, for iterations $i = 1 : (M+1) - M^{'}$ which is an $M \times ((M+1) - (i-1))$ matrix , which by the standard "Golub-Kahan-Reinsch" SVD algorithm takes, see Chapter 8.6.4 [106]*

$$4M^2 \times (M+1-(i-1)) + 8M \times (M+1-(i-1))^2 + 9 \times (M+1-(i-1))^3 =$$

$$21M^3 + \textit{lower order terms}$$

If $M^{'} = M$, then, we perform only one iteration through steps $3) - 8)$and hence, we have $C(\eta_{M+1}, \tilde{\eta}_{M'}) = O(M^3)$. If $M^{'} < M$ then the number of iterations through steps $3) - 8)$ is greater than one, however it is generally much smaller than M, we will assume that $C(\eta_{M+1}, \tilde{\eta}_{M'}) = O(M^3)$.

Algorithm 5: Recombination Algorithm with 1-Tree Measure Reduction

Input: Let $\nu = \sum_{i=1}^{N} \omega_i \delta_{\underline{\boldsymbol{x}}_i}$ be an atomic probability measure on $\mathbb{R}^d$, with $supp(\nu) \subseteq K$, $\Psi = (\psi_1, \ldots, \psi_M) : \ K \to \mathbb{R}^M$ be a continuos, ν-integrable map

1) Produce the push-forward of ν through Ψ

$$(\Psi_* \nu) = \sum_{i=1}^{N} \omega_i \delta_{\Psi(\underline{\boldsymbol{x}}_i)}$$

2) Produce, via RHFC Algorithm, an $(M+1)$-Tree form of $(\Psi_* \nu)$

$$\eta_{M+1}^0 = \sum_{i=1}^{M+1} \beta_i^0 \delta_{\underline{\boldsymbol{y}}_i^0} \qquad \beta_i^0 = P(T_i^0), \ \underline{\boldsymbol{y}}_i^0 = R(T_i^0)$$

and set $j = 0$.

3) Produce, via 1-Tree Measure Reduction Algorithm and tree updating, a reduced tree measure η_M^j for η_{M+1}^j

$$\eta_M^j = \sum_{i=1}^{M} \widehat{\beta}_k^j \delta_{\underline{\boldsymbol{y}}_{i_k}^j} \qquad \widehat{\beta}_k^j = P(\widehat{T}_{i_k}^j), \ \underline{\boldsymbol{y}}_{i_k}^j = R(\widehat{T}_{i_k}^j)$$

where each tree $\widehat{T}_{i_k}^j$ is a weight-updated tree $T_{i_k}^j$, corresponding to the root weight update $\beta_k^j \to \widehat{\beta}_k^j$

If $\text{depth}(\widehat{T}_{i_k}^j) > 1$, for some $k \in [\![1, M]\!]$, proceed to step 4), else go to step 6)

4) Select the binary tree of greatest depth $\widehat{T}_{i_r}^j \in \left\{\widehat{T}_{i_1}^j, \ldots, \widehat{T}_{i_M}^j\right\}$ and split the tree into two subtrees, by removing its root node

$$\left(P(\widehat{T}_{i_r}^j), \ R(\widehat{T}_{i_r}^j)\right)$$

$$\left(P(\widehat{T}_{i_{r_1}}^j), \ R(\widehat{T}_{i_{r_1}}^j)\right) \qquad \left(P(\widehat{T}_{i_{r_2}}^j), \ R(\widehat{T}_{i_{r_2}}^j)\right)$$

5) Set $\{T_1^{j+1},\ldots,T_{M+1}^{j+1}\} := \left(\{\widehat{T}_{i_1}^j,\ldots,\widehat{T}_{i_M}^j\}\backslash\{\widehat{T}_{i_r}^j\}\right)\bigcup\{\widehat{T}_{i_{r_1}}^j,\widehat{T}_{i_{r_2}}^j\}$ and produce $(M+1)$-Tree form of $(\Psi_*\nu)$, given by

$$\eta_{M+1}^{j+1} = \sum_{i=1}^{M+1}\beta_i^{j+1}\delta_{\underline{\boldsymbol{y}}_i^{j+1}} \qquad \beta_i^{j+1} = P(T_i^{j+1}),\ \underline{\boldsymbol{y}}_i^{j+1} = R(T_i^{j+1})$$

set $j = j+1$ and go to step 3).

6) Denote the output measure from step 3) by

$$\widehat{(\Psi_*\nu)} = \sum_{k=1}^{M}\widehat{\omega}_k\delta_{\Psi(\underline{\boldsymbol{x}}_{i_k})}$$

7) **Output**: Ψ-generalised Carathéodory cubature measure for ν:

$$\widehat{\nu}_\Psi = \sum_{k=1}^{M}\widehat{\omega}_k\delta_{\underline{\boldsymbol{x}}_{i_k}}$$

Remark 3.2.2. *In the preceding algorithm: Recombination Algorithm with* 1*-Tree Measure Reduction, we have assumed that each measure* η_{M+1}^j*, generating a coefficient matrix* $A_j^{(0)} := [\underline{\boldsymbol{y}}_1,\underline{\boldsymbol{y}}_2,\ldots,\underline{\boldsymbol{y}}_{M+1}]$ *has rank*$(A_j^{(0)})$*=*$M' = M$*, as described in* 1*-Tree Measure Reduction, for simplicity of exposition.*

3.2.0.11 Theoretical Complexity of the Recombination Algorithm with 1-Tree Measure Reduction

The computational bottleneck of the Recombination Algorithm consists of the inherently sequential 1-Tree Measure Reduction in step and Tree Splitting operations in steps 3)-5), which consume over 90% of the algorithms runtime. An implementation of steps 3)-5) was given in [96]. Let us now examine the overall complexity.

In step 1), we produce the push-forward of $\nu = \sum_{i=1}^{N} \omega_i \delta_{\underline{\boldsymbol{x}}_i}$ with $supp(\nu) \subseteq \mathbb{R}^d$ through the M-dimensional map $\Psi = (\psi_1, \ldots, \psi_M)$, yielding

$$(\Psi_* \nu) = \sum_{i=1}^{N} \omega_i \delta_{\Psi(\underline{\boldsymbol{x}}_i)} \text{ with } supp(\Psi_* \nu) \subseteq \mathbb{R}^M$$

which by construction has a centre of mass, satisfying

$$\begin{aligned} \mathcal{C}oM\big(\Psi_* \nu\big) &= \Big[\sum_{i=1}^{N} \omega_i \psi_1(\underline{\boldsymbol{x}}_i), \ldots, \sum_{i=1}^{N} \omega_i \psi_M(\underline{\boldsymbol{x}}_i) \Big]^T \\ &= \Big[\int_K \psi_1(\underline{\boldsymbol{x}}) d\nu, \ldots, \int_K \psi_M(\underline{\boldsymbol{x}}) d\nu \Big]^T \end{aligned}$$

Thus, we have transformed out original goal of constructing a Ψ-generalised Carathéodory cubature measure for ν: $\widehat{\nu}_\Psi = \sum_{k=1}^{M} \widehat{\omega}_k \delta_{\underline{\boldsymbol{x}}_{i_k}}$, that is a measure on $\mathbb{R}^d$ satisfying

$$\begin{aligned} &1) \ \int_{\mathbb{R}^d} \psi_j(\underline{\boldsymbol{x}}) d\nu(\underline{\boldsymbol{x}}) = \int_{\mathbb{R}^d} \psi_j(\underline{\boldsymbol{x}}) d\widehat{\nu}_\Psi(\underline{\boldsymbol{x}}) \quad \forall j \in [\![1, M]\!] \\ &2) \ supp(\widehat{\nu}_\Psi) \subseteq supp(\nu) \\ &3) \ card\big(supp(\widehat{\nu}_\Psi^\mu)\big) \leq M + 1 \end{aligned} \tag{3.12}$$

into constructing a reduced measure $(\widehat{\Psi_* \nu}) = \sum_{k=1}^{M} \widehat{\omega}_k \delta_{\Psi(\underline{\boldsymbol{x}}_{i_k})}$ for $(\Psi_* \nu)$, on $\mathbb{R}^M$, satisfying

$$\begin{aligned} &1) \ \mathcal{C}oM(\Psi_* \nu) = \sum_{k=1}^{N} \omega_i \Psi(\underline{\boldsymbol{x}}_i) = \sum_{i=1}^{M} \widehat{\omega}_k \Psi(\underline{\boldsymbol{x}}_{i_k}) = \mathcal{C}oM(\widehat{\Psi_* \nu}) \\ &2) \ supp(\widehat{\Psi_* \nu}) \subseteq supp(\Psi_* \nu) \\ &3) \ card(supp(\widehat{\Psi_* \nu})) \leq M + 1 \end{aligned}$$

In step 1) the cost of producing the push-forward of ν through the map $\Psi := (\psi_1, \ldots, \psi_M)$ is dMN, since we are evaluating an M-dimensional map at N d-dimensional points.

In step 2) we proceed to cluster the measure $(\Psi_* \nu)$, via RHFC Algorithm. For simplicity of exposition of step 2), we assume that $N = 2^S$ and $(M + 1) = 2^P$ for some $S > P$. Then, η_{M+1}^0, produced in step 2), is given by

$$\eta_{M+1}^0 = \sum_{i=1}^{M+1} \beta_i^0 \delta_{\underline{\boldsymbol{y}}_i^0} : \quad \beta_i^0 = P(T_i^0), \quad \underline{\boldsymbol{y}}_i^0 = R(T_i^0) \quad \text{for} \quad 1 \leq i \leq M + 1$$

where each tree in the binary forest $\{T_1^0, \ldots, T_{M+1}^0\}$ has $depth(T_i^0) = \lceil log_2(\frac{2^S}{2^P}) \rceil = (S-P)$ and an associated collection of leaf nodes of cardinality $card(l(T_i^0)) = 2^{S-P}$.

On the first pass through steps 3)-5) of the Recombination Algorithm, 1-Tree Measure Reduction in step 3) eliminates one binary tree of depth $(S-P)$ from $\{T_1^0, \ldots, T_{M+1}^0\}$, thereby eliminating $2^{(S-P)}$ points from the $supp(\nu)$, the support of the input measure. Clearly, on the subsequent passes through the algorithm, we have no control over the depth of the binary tree eliminated by the 1-Tree Measure Reduction. However, we can derive the upper and lower bounds on the number of iterations, through the Recombination Algorithm, required to halve the support of ν.

The fastest procedure to halve the support of ν, is for the 1-Tree Measure Reduction to eliminate all trees of depth $(S-P)$, which takes at least $\frac{(M+1)}{2}$ iterations and results in the reduction of $supp(\nu)$ by $\frac{(M+1)}{2} \cdot 2^{(S-P)} = 2^S/2$ points.

Whilst, the slowest procedure to halve the support of ν, is as follows: on the first pass, the algorithm always eliminates one tree of depth $(S-P)$ and splits another tree of depth $(S-P)$ to produce two trees of depth $(S-P)-1$. Then, since at step 4), the algorithm always splits the tree of greatest depth, it would take at most M iterations to reduce all the trees of depth $(S-P)$ to depth $(S-P)-1$. In which case, if the 1-Tree Measure Reduction always eliminates a tree of depth $(S-P)-1$ on iterations $i = 2:M$, then, after M iterations the $supp(\nu)$ is reduced by

$$\begin{aligned} & 1 \cdot 2^{(S-P)} + (M-1) \cdot 2^{(S-P)-1} \\ = \; & 1 \cdot 2^{(S-P)} + \Big(\frac{M+1}{2} - 1\Big) \cdot 2^{(S-P)} \\ = \; & \frac{(M+1)}{2} \cdot 2^{(S-P)} = 2^S/2 \end{aligned}$$

Hence, it takes at least $(M+1)/2$ and at most M iterations through steps 3)-5) of the Recombination Algorithm with 1-Tree Measure Reduction to halve the $supp(\nu)$. Hence, the total number of iterations through steps 3)-5) of the Recombination Algorithm is of order $O(Mlog\big(N/(M+1)\big))$.

The computational complexity of the RHFC algorithm in step 2) can be computed as follows. Starting with the level $j = 0$ of the forthcoming binary forest $\{T_1, \ldots, T_{M+1}\}$, containing N nodes, the formation of each successive level $j \in [\![1, D]\!]$, $D := \lceil log_2(\frac{N}{L}) \rceil$

requires $\frac{N}{2^j}$ multiplications by a scalar and additions of two nodes of length M, hence adding over all the levels we get

$$2M\sum_{j=1}^{D}\frac{N}{2^j} = 2NM\big[1-\frac{1}{4}(\frac{1}{2})^D\big] \leq \frac{7}{4}NM$$

Now, let $C(\eta_{M+1}^j, \tilde{\eta}_M^j)$ denote the complexity of performing a 1-Tree Measure Reduction and let $C_{up}(\tilde{\eta}_M^j, \eta_M^j)$ denotes the complexity of updating the weights down each tree $T_{i_k} \in \{T_{i_1}, \ldots, T_{i_M}\}$ in the $supp(\tilde{\eta}_M^j)$. Then, the computational complexity of the Recombination Algorithm is given by

$$O\Big(NM + Mlog\big(N/(M+1)\big)\big[C(\eta_{M+1}^j, \tilde{\eta}_M^j) + C_{up}(\tilde{\eta}_M^j, \eta_M^j)\big]\Big) \tag{3.13}$$

where the second term arises from $O(Mlog\big(N/(M+1)\big))$ iterations through steps 3)-5). We already know that, $C(\eta_{M+1}^j, \tilde{\eta}_M^j) = O(M^3)$ and as stated in Section 3.1.0.7, we can produce a weight-updated tree $\widehat{T}_{i_k}$ from T_{i_k} in $O(1)$, by employing trivial parallelisation, and therefore we have $C_{up}(\tilde{\eta}_M^j, \eta_M^j) = O(M)$. Moreover, we note that the tree splitting procedure in step 4) simply requires referencing the left and right subtrees of the tree of greatest depth, rather than the tree itself and thus can be accomplished in order $O(1)$.

Hence, the cost of the Recombination Algorithm with 1-Tree Measure Reduction is given by

$$O\Big(NM + log\big(N/(M+1)\big)M^4\Big) \tag{3.14}$$

In the following chapter,we will construct a further recursion, that will enable us to construct the RHFC Algorithm for $N \approx O(M^2)$, hence we may assume that the overall cost is dominated by the term $log\big(N/(M+1)\big)M^4$.

3.2.0.12 Updating the Recombination Algorithm with 1-Tree Measure Reduction

Conceptually, it is easy to introduce an updating scheme to improve the overall performance of the Recombination Algorithm if we assume that η_{M+1} produced in steps 2) and 5) always generates a coefficient matrix of full rank, as illustrated in the following.

We assume that $\eta^0_{M+1} = \sum_{j=1}^{M+1} \beta^0_j \delta_{\underline{\boldsymbol{y}}^0_j}$ is produced in step 2) and set

$$A^{(0)}_0 := [\underline{\boldsymbol{y}}^0_1, \underline{\boldsymbol{y}}^0_2, \ldots, \underline{\boldsymbol{y}}^0_{M+1}] \in \mathbb{R}^{M\times(M+1)},\ rank(A^{(0)}_0) = M$$
$$\underline{\boldsymbol{\beta}}^0 := [\beta^0_1, \beta^0_2, \ldots, \beta^0_{M+1}]^T \in \mathbb{R}^{(M+1)}_+,\ \sum_{j=1}^{M+1} \beta^0_j = 1$$

Now, let N_t denote the total number of iterations through steps $3) - 5)$ of the Recombination Algorithm with 1-Tree Measure Reduction.

Then for $1 \le j \le N_t$, in in step 3) of the Recombination Algorithm, we have

$$\eta^j_{M+1} = \sum_{\ell=1}^{M+1} \beta^j_\ell \delta_{\underline{\boldsymbol{y}}^j_\ell}, \qquad \beta^j_\ell = P(T^j_\ell),\ \underline{\boldsymbol{y}}^j_\ell = R(T^j_\ell)$$

and we set

$$A^{(0)}_j := [\underline{\boldsymbol{y}}^j_1, \underline{\boldsymbol{y}}^j_2, \ldots, \underline{\boldsymbol{y}}^j_{M+1}] \in \mathbb{R}^{M\times(M+1)},\ rank(A^{(0)}_j) = M$$
$$\underline{\boldsymbol{\beta}}^j := [\beta^j_1, \beta^j_2, \ldots, \beta^j_{M+1}]^T \in \mathbb{R}^{(M+1)}_+,\ \sum_{j=1}^{M+1} \beta^j_i = 1$$

Then, $dim(\mathcal{K}er(A^{(0)}_j)) = 1$ and let us assume that $\mathcal{K}er(A^{(0)}_j) = span\{\underline{\boldsymbol{\phi}}^j\}$, then we proceed with the 1-Tree Measure Reduction Algorithm, yielding

$$\alpha^j_{(1)} := \min_{1\le\ell\le M+1} \left\{ \frac{\beta^j_\ell}{\phi^j_\ell} : \phi^j_\ell > 0 \right\} = \frac{\beta^j_k}{\phi^j_k} \quad \text{for some } k \in [\![1, M+1]\!]$$

and set

$$\widehat{\beta}^j_{i_\ell} := \begin{cases} \beta^j_\ell - \alpha^j_{(1)}\phi^j_\ell & \text{for} \quad \ell \in [\![1, k-1]\!] \\ \beta^j_{\ell+1} - \alpha^j_{(1)}\phi^j_{\ell+1} & \text{for} \quad \ell \in [\![k, M]\!] \end{cases}$$

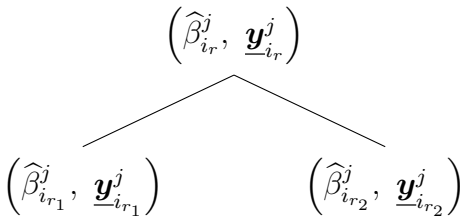
$$\underline{\boldsymbol{y}}^j_{i_\ell} := \begin{cases} \underline{\boldsymbol{y}}^j_\ell & \text{for} \quad \ell \in [\![1, k-1]\!] \\ \underline{\boldsymbol{y}}^j_{\ell+1} & \text{for} \quad \ell \in [\![k, M]\!] \end{cases}$$

and whence produce a reduced tree measure η^j_M for η^j_{M+1}

$$\eta^j_M = \sum_{i=1}^{M} \widehat{\beta}^j_{i_\ell} \delta_{\underline{\boldsymbol{y}}^j_{i_\ell}} \qquad \widehat{\beta}^j_\ell = P(\widehat{T}^j_{i_\ell}),\ \underline{\boldsymbol{y}}^j_{i_\ell} = R(\widehat{T}^j_{i_\ell})$$

where each tree $\widehat{T}^j_{i_\ell}$ is a weight-updated tree $T^j_{i_\ell}$, corresponding to the root weight update $\beta^j_{i_\ell} \to \widehat{\beta}^j_{i_\ell}$ and the tree weight update is performed, as in Section 3.1.0.7.

Proceeding to step 4) of the Recombination Algorithm, we select the binary tree of greatest depth $\widehat{T}^j_{i_r} \in \{\widehat{T}^j_{i_1}, \ldots, \widehat{T}^j_{i_M}\}$ and split it into two sub-trees $\{\widehat{T}^j_{i_{r_1}}, \widehat{T}^j_{i_{r_2}}\}$ of depth $depth(\widehat{T}^j_{i_r}) - 1$, which corresponds to



Without loss of generality, we assume that $r < k$ and define

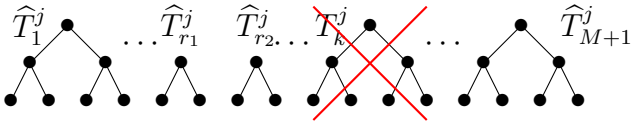
$$\beta^{j+1}_\ell := \begin{cases} \widehat{\beta}^j_{i_\ell} & \text{for } \ell \in [\![1, r-1]\!] \\ \widehat{\beta}^j_{i_{r_1}} & \text{for } \ell = r \\ \widehat{\beta}^j_{i_\ell} & \text{for } \ell \in [\![r+1, k-1]\!] \\ \widehat{\beta}^j_{i_{r_2}} & \text{for } \ell = k \\ \widehat{\beta}^j_{i_{\ell-1}} & \text{for } \ell \in [\![k+1, M+1]\!] \end{cases}$$

$$\underline{\boldsymbol{y}}^{j+1}_\ell := \begin{cases} \underline{\boldsymbol{y}}^j_{i_\ell} & \text{for } \ell \in [\![1, r-1]\!] \\ \underline{\boldsymbol{y}}^j_{i_{r_1}} & \text{for } \ell = r \\ \underline{\boldsymbol{y}}^j_{i_\ell} & \text{for } \ell \in [\![r+1, k-1]\!] \\ \underline{\boldsymbol{y}}^j_{i_{r_2}} & \text{for } \ell = k \\ \underline{\boldsymbol{y}}^j_{i_{\ell-1}} & \text{for } \ell \in [\![k+1, M+1]\!] \end{cases}$$

Hence, in step 5) of the Recombination Algorithm, we obtain

$$\eta^{j+1}_{M+1} = \sum_{\ell=1}^{M+1} \beta^{j+1}_\ell \delta_{\underline{\boldsymbol{y}}^{j+1}_\ell}, \quad \text{where}$$
$$\beta^{j+1}_\ell = P(\widehat{T}^j_\ell), \ \ \underline{\boldsymbol{y}}^{j+1}_\ell = R(\widehat{T}^j_\ell) \ \text{ for } \ 1 \le j \le M+1, \ j \ne r, \ k$$
$$\beta^{j+1}_r = P(\widehat{T}^j_{r_1}), \ \ \underline{\boldsymbol{y}}^{j+1}_r = R(\widehat{T}^j_{r_1}), \ \beta^{j+1}_k = P(\widehat{T}^j_{r_2}), \ \ \underline{\boldsymbol{y}}^{j+1}_k = R(\widehat{T}^j_{r_2})$$

which pictorially corresponds to



$$A_j^{(0)} = [\underline{\boldsymbol{y}}_1^j, \ldots, \underline{\boldsymbol{y}}_r^j, \ldots, \underline{\boldsymbol{y}}_k^j, \ldots, \underline{\boldsymbol{y}}_{M+1}^j]$$
$$A_{j+1}^{(0)} = [\underline{\boldsymbol{y}}_1^j, \ldots, \underline{\boldsymbol{y}}_{r_1}^j, \ldots, \underline{\boldsymbol{y}}_{r_2}^j, \ldots, \underline{\boldsymbol{y}}_{M+1}^j]$$

Hence, we have

$$\left(A_{j+1}^{(0)} - A_j^{(0)}\right) = [0, \ldots, 0, (\underline{\boldsymbol{y}}_{r_1}^j - \underline{\boldsymbol{y}}_r^j), 0, \ldots, 0, (\underline{\boldsymbol{y}}_{r_2}^j - \underline{\boldsymbol{y}}_k^j), 0, \ldots, 0] = \Delta A_j^{(0)} \qquad (3.15)$$

Therefore, only two columns of the underlying coefficient matrix $A_j^{(0)}$ are altered at the jth iteration through the Recombination Algorithm and therefore the increment $\Delta A_j^{(0)} = \Delta A_j^{(0,1)} + \Delta A_j^{(0,2)}$ can be represented as a sum of two rank-one matrices

$$\Delta A_j^{(0,1)} = \underline{\boldsymbol{p}}^{(j,1)} \left[\underline{\boldsymbol{q}}^{(j,1)}\right]^T, \qquad \Delta A_j^{(0,2)} = \underline{\boldsymbol{p}}^{(j,2)} \left[\underline{\boldsymbol{q}}^{(j,2)}\right]^T$$

$$\underline{\boldsymbol{p}}^{(j,1)} = (\underline{\boldsymbol{y}}_{r_1}^j - \underline{\boldsymbol{y}}_r^j), \quad \underline{\boldsymbol{q}}^{(j,1)} = \underline{\boldsymbol{e}}_r$$
$$\underline{\boldsymbol{p}}^{(j,2)} = (\underline{\boldsymbol{y}}_{r_2}^j - \underline{\boldsymbol{y}}_k^j), \quad \underline{\boldsymbol{q}}^{(j,2)} = \underline{\boldsymbol{e}}_k$$

f It would be desirable to exploit the aforementioned relationship between two consecutive coefficient matrices $A_j^{(0)}$ and $A_{j+1}^{(0)}$ and obtain an efficient updating method of the underlying coefficient matrix, that allows us to track the kernel throughout the iterative cycle of the Recombination Algorithm. Updating matrix factorisations stably and efficiently has been the subject of extensive research in numerical linear algebra and signal processing. Gill, Golub, Murray and Saunders published a landmark paper on modifying matrix factorisations in [79], and Stewart proposed several matrix decompositions for updating algorithms in subspace tracking [80], [81].

In our context, a potential method for exploiting the relationship (3.15), is to utilise the QR-decomposition, in place of the SVD to compute the kernel of $A_j^{(0)}$, as it is amenable to updates as opposed to the SVD. Consider the following updating scheme

1. Compute QR decomposition of $[A_0^{(0)}]^T$

$$[A_0^{(0)}]^T = \big[Q_1^{(0)} \ \underline{\boldsymbol{q}_2}^{(0)}\big] \begin{bmatrix} R_1^{(0)} \\ 0 \end{bmatrix} = Q^{(0)} R^{(0)}$$

where $span\{\underline{\boldsymbol{q}_2}^{(0)}\} = \big(\mathcal{K}er(A_0^{(0)})\big)$, set $\underline{\boldsymbol{\phi}}^0 := \underline{\boldsymbol{q}_2}^{(0)}$

For $j = 0 : N_t - 1$
For $i = 1, 2$

2. Compute $\underline{\boldsymbol{p}}^{(j,i)}\big[\underline{\boldsymbol{q}}^{(j,i)}\big]^T$ and update the QR decomposition of $A_j^{(0)}$ to $A_{j+1}^{(0)}$

$$\Big[A_{j+1}^{(0)}\Big]^T \ \rightarrow \ \Big[A_j^{(0)} + \underline{\boldsymbol{p}}^{(j,i)}\big[\underline{\boldsymbol{q}}^{(j,i)}\big]^T\Big]^T$$

3. From QR decomposition of $[A_{j+1}^{(0)}]^T$

$$[A_{j+1}^{(0)}]^T = \big[Q_1^{(j+1)} \ \underline{\boldsymbol{q}_2}^{(j+1)}\big] \begin{bmatrix} R_1^{(j+1)} \\ 0 \end{bmatrix} = Q^{(j+1)} R^{(j+1)}$$

and set $\underline{\boldsymbol{\phi}}^j := \underline{\boldsymbol{q}_2}^{(j)}$

We recall that the total cost of the Recombination Algorithm with 1-Tree Measure Reduction is given by

$$O\Big(NM + Mlog\big(N/(M+1)\big)\big[C(\eta_{M+1}^j, \tilde{\eta}_M^j) + C_{up}(\tilde{\eta}_M^j, \eta_M^j)\big]\Big)$$

where $C(\eta_{M+1}^j, \tilde{\eta}_M^j)$ denotes the complexity of performing a 1-Tree Measure Reduction, dominated by the computation of the $\mathcal{K}er(A_j^{(0)})$, via SVD, at each iteration $0 \leq j \leq N_t$ and $C_{up}(\tilde{\eta}_M^j, \eta_M^j)$ denotes the complexity of updating the weights down each tree $T_{i_k} \in \{T_{i_1}, \ldots, T_{i_M}\}$ in the $supp(\tilde{\eta}_M^j)$, yielding η_M^j.

By exploiting the relationship (3.15) between two consecutive systems $A_j^{(0)}$ and $A_{j+1}^{(0)}$ and introducing the aforementioned updating scheme $A_j^{(0)} \rightarrow A_{j+1}^{(0)}$, let us denote by $C(\eta_{M+1}^0, \tilde{\eta}_M^0)$ the complexity to perform the 1^{st} 1-Tree Measure Reduction and let $C(\eta_{M+1}^j, \tilde{\eta}_M^j)$ denote the complexity of performing the subsequent 1-Tree Measure Reductions for $1 \leq j \leq N_t$, then we can conclude that the total cost of the Recombination Algorithm is given by

$$O\Big(NM + \big[C(\eta_{M+1}^0, \tilde{\eta}_M^0) + C(\tilde{\eta}_M^0, \eta_M^0)\big] + Mlog\big(N/(M+1)\big)\big[C(\eta_{M+1}^j, \tilde{\eta}_M^j) + C_{up}(\tilde{\eta}_M^j, \eta_M^j)\big]\Big)$$

As illustrated in the aforementioned updating scheme, $C(\eta^0_{M+1}, \tilde{\eta}^0_M)$ is dominated by the computation of the QR decomposition of $A^{(0)}_0$, and hence $C(\eta^0_{M+1}, \tilde{\eta}^0_M) = O(M^3)$. Whilst the cost of the subsequent 1-Tree Measure Reductions $C(\eta^j_{M+1}, \tilde{\eta}^j_M)$, will be dominated by the computation of two rank-1 updates of the QR decomposition, and therefore we have $C(\eta^j_{M+1}, \tilde{\eta}^j_M) = O(M^2)$, as noted in Chapter 6.5 of [104] and $C_{up}(\tilde{\eta}^j_M, \eta^j_M) = O(M)$. Hence, the overall cost of the Recombination Algorithm with the updating scheme is given by

$$O\Big(NM + \big[1 + log\big(N/(M+1)\big]M^3\Big)$$

We note that even if the aforementioned method of updating the underlying systems of linear equations from one iteration to the next, proves to be numerically stable, it still does not address the issue of rank-degenracy of the underlying matrix $A^{(0)}_j$. Clearly, in the context of a rank deficient matrix $A^{(0)}_j$, the updating scheme described above would not work, as we would not know the $rank(A^{(0)}_j)$ and hence $dim(\mathcal{K}er(A^{(0)}_j))$ in advance, and whilst it is possible to modify the QR factorisation so as to obtain a *Rank-Revealing* QR Factorisation, see [113], such a factorisation is not amenable to updates. Hence, overall we must conclude that the idea of updating the 1-Tree Measure Reduction is not a feasible method for reducing the complexity of the Recombination Algorithm.

In the following section, we introduce a new measure reduction algorithm, that achieves the same reduction in complexity of the overall Recombination Algorithm, as the updating scheme. Furthermore, it does not yield the same numerical instabilities and works even when the underlying systems of equations exhibit rank-degeneracies.

3.3 M-Tree Measure Reduction Algorithm

Algorithm 6: M-Tree Measure Reduction

Input: Let η_{2M} be an atomic probability measure on $\mathbb{R}^M$, given by

$$\eta_{2M} = \sum_{i=1}^{2M} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$$

1) Set

$$A^{(0)} := [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_{2M}]$$

$$\underline{\boldsymbol{\beta}}^{(0)} := [\beta_1, \beta_2, \ldots, \beta_{2M}]^T$$

2) Compute $\mathcal{K}er(A^{(0)}) = \mathcal{C}ol(\Phi^{(0)})$

$$\Phi^{(0)} := [\underline{\boldsymbol{\phi}}_1^{(0)}, \underline{\boldsymbol{\phi}}_2^{(0)}, \ldots, \underline{\boldsymbol{\phi}}_M^{(0)}]$$

For $i = 1 : M$

3) Select $\underline{\boldsymbol{\phi}}_1^{(i-1)}$ from $\Phi^{(i-1)}$ and compute $\alpha_{(i)}$

$$\alpha_{(i)} := \min_{1 \le j \le 2M-(i-1)} \left\{ \frac{\beta_j^{(i-1)}}{(\underline{\boldsymbol{\phi}}_1^{(i-1)})_j} : (\underline{\boldsymbol{\phi}}_1^{(i-1)})_j > 0 \right\} = \frac{\beta_{k^{(i)}}^{(i-1)}}{(\underline{\boldsymbol{\phi}}_1^{(i-1)})_{k^{(i)}}}$$

for some $k^{(i)} \in [\![1, 2M-(i-1)]\!]$

4) Set $\underline{\boldsymbol{\beta}}^{(i)} := [\beta_1^{(i)}, \beta_2^{(i)}, \ldots, \beta_{2M-i}^{(i)}]^T$

$$\beta_j^{(i)} := \begin{cases} \beta_j^{(i-1)} - \alpha_{(i)} (\underline{\boldsymbol{\phi}}_1^{(i-1)})_j & \text{for } j \in [\![1, k^{(i)} - 1]\!] \\ \beta_{j+1}^{(i-1)} - \alpha_{(i)} (\underline{\boldsymbol{\phi}}_1^{(i-1)})_{j+1} & \text{for } j \in [\![k^{(i)}, 2M - i]\!] \end{cases}$$

5) Set $A^{(i)} := [\underline{\boldsymbol{y}}_1^{(i)}, \underline{\boldsymbol{y}}_2^{(i)}, \ldots, \underline{\boldsymbol{y}}_{2M-i}^{(i)}]$

$$\underline{\boldsymbol{y}}_j^{(i)} := \begin{cases} \underline{\boldsymbol{y}}_j^{(i-1)} & \text{for } j \in [\![1, k^{(i)} - 1]\!] \\ \underline{\boldsymbol{y}}_{j+1}^{(i-1)} & \text{for } j \in [\![k^{(i)}, 2M - i]\!] \end{cases}$$

6) Compute $d_{\ell+1}^{(i)}$

$$d_{\ell+1}^{(i)} := \frac{(\underline{\boldsymbol{\phi}}_{\ell+1}^{(i-1)})_{k^{(i)}}}{(\underline{\boldsymbol{\phi}}_{1}^{(i-1)})_{k^{(i)}}} \quad \text{for } \ell \in [\![1, M-i]\!]$$

7) Compute

$$\widehat{\underline{\boldsymbol{\phi}}}_{\ell}^{(i)} := \underline{\boldsymbol{\phi}}_{\ell+1}^{(i-1)} - d_{\ell+1}^{(i)} \underline{\boldsymbol{\phi}}_{1}^{(i-1)} \quad \text{for } \ell \in [\![1, M-i]\!]$$

8) Produce $\Psi^{(i)} = [\underline{\boldsymbol{\phi}}_{1}^{(i)}, \underline{\boldsymbol{\phi}}_{2}^{(i)}, \ldots, \underline{\boldsymbol{\phi}}_{M-i}^{(i)}]$, by computing

$$(\underline{\boldsymbol{\phi}}_{\ell}^{(i)})_j := \begin{cases} (\widehat{\underline{\boldsymbol{\phi}}}_{\ell}^{(i)})_j & \text{for } j \in [\![1, k^{(i)}-1]\!] \\ (\widehat{\underline{\boldsymbol{\phi}}}_{\ell}^{(i)})_{j+1} & \text{for } j \in [\![k^{(i)}, 2M-i]\!] \end{cases}$$

end i **loop**

9) Set $\hat{\underline{\boldsymbol{\beta}}} = [\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_M]^T := [\beta_1^{(M)}, \beta_2^{(M)}, \ldots, \beta_M^{(M)}]^T = \underline{\boldsymbol{\beta}}^{(M)}$

10) Set $\hat{A} = [\underline{\boldsymbol{y}}_{i_1}, \underline{\boldsymbol{y}}_{i_2}, \ldots, \underline{\boldsymbol{y}}_{i_M}] := [\underline{\boldsymbol{y}}_1^{(M)}, \underline{\boldsymbol{y}}_2^{(M)}, \ldots, \underline{\boldsymbol{y}}_M^{(M)}] = A^{(M)}$

Output: Reduced measure for η_{2M}

$$\tilde{\eta}_M = \sum_{k=1}^{M} \hat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}}$$

The following provides a constructive proof of lemma (3.0.28) for $L = 2M$.

Proof. Given $\eta_{2M} = \sum_{i=1}^{2M} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$ an atomic probability measure on $\mathbb{R}^M$. Set

$$\underline{\boldsymbol{y}}_i = [1, y_{2,i} \ldots, y_{M,i}]^T \in \mathbb{R}^M$$
$$A^{(0)} := [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_{2M}] \in \mathbb{R}^{M \times 2M}$$
$$\underline{\boldsymbol{\beta}}^{(0)} := [\beta_1, \beta_2, \ldots, \beta_{2M}]^T \in \mathbb{R}_+^{2M}, \ \sum_{i=1}^{2M} \beta_i = 1$$

For simplicity of exposition, we assume $A^{(0)}$ is full-ranked, we address the rank-deficient case, which will require a modification of the M-Tree Measure Reduction Algorithm in section 3.3.1. Since $rank(A^{(0)}) = M$, then $dim(\mathcal{K}er(A^{(0)})) = M$ and by step 2) we compute $\mathcal{K}er(A^{(0)}) = Col(\Phi^{(0)})$, where

$$\Phi^{(0)} := [\underline{\boldsymbol{\phi}}_{1}^{(0)}, \underline{\boldsymbol{\phi}}_{2}^{(0)}, \ldots, \underline{\boldsymbol{\phi}}_{M}^{(0)}] \in \mathbb{R}^{2M \times M}$$

Clearly, since $A^{(0)}\underline{\boldsymbol{\phi}}_i^{(0)} = \underline{\mathbf{0}}$, $\forall i \in [\![1, M]\!]$ we have

$$\sum_{j=1}^{2M} (\underline{\boldsymbol{\phi}}_i^{(0)})_j \underline{\boldsymbol{y}}_j = \underline{\mathbf{0}}$$

and in particular since $y_{1,j} = 1$, $\forall j \in [\![1, 2M]\!]$, we have

$$\sum_{j=1}^{2M} (\underline{\boldsymbol{\phi}}_i^{(0)})_j = 0$$

hence, for each $i \in [\![1, M]\!]$ at least one $(\underline{\boldsymbol{\phi}}_i^{(0)})_j > 0$, as not all $(\underline{\boldsymbol{\phi}}_i^{(0)})_j = 0$

$$\mathcal{C}oM(\eta_{2M}) = \sum_{j=1}^{2M} \beta_j \underline{\boldsymbol{y}}_j = \sum_{j=1}^{2M} \beta_j \underline{\boldsymbol{y}}_j - \alpha \left(\sum_{j=1}^{2M} (\underline{\boldsymbol{\phi}}_i^{(0)})_j \underline{\boldsymbol{y}}_j \right) = \sum_{j=1}^{2M} \left(\beta_j - \alpha (\underline{\boldsymbol{\phi}}_i^{(0)})_j \right) \underline{\boldsymbol{y}}_j$$

holds for any $\alpha \in \mathbb{R}$ and in particular for $i = 1$ we choose $\alpha = \alpha_{(1)}$, defined in step 3), and $\underline{\boldsymbol{\phi}}_i^{(0)} = \underline{\boldsymbol{\phi}}_1^{(0)}$ then we have $\beta_j - \alpha_{(1)}(\underline{\boldsymbol{\phi}}_1^{(0)})_j \geq 0$, $\forall j \in [\![1, 2M]\!]$, $\beta_{k^{(1)}} - \alpha_{(1)}(\underline{\boldsymbol{\phi}}_1^{(0)})_{k^{(1)}} = 0$ for some $k^{(1)} \in [\![1, 2M]\!]$ and $\sum_{j=1}^{2M} \left(\beta_j - \alpha_{(1)}(\underline{\boldsymbol{\phi}}_1^{(0)})_j \right) = 1$.

Hence, we have produced a probability vector $\left[\underline{\boldsymbol{\beta}}^{(0)} - \alpha_{(1)} \underline{\boldsymbol{\phi}}_1^{(0)} \right]$ placing zero mass on the $k^{(1)}th$ node in $supp(\eta_{2M})$. Subsequently, we shrink the probability vector produced by removing the zeroed coordinate to obtain

$$\underline{\boldsymbol{\beta}}^{(1)} := [\beta_1^{(1)}, \beta_2^{(1)}, \ldots, \beta_{2M-1}^{(1)}]^T \in \mathbb{R}_+^{2M-1}, \ \sum_{j=1}^{2M-1} \beta_j^{(1)} = 1$$

as defined in step 4). And since the $k^{(1)}th$ node $\underline{\boldsymbol{y}}_{k^{(1)}}$ can be represented as a linear combination of $\left\{ \underline{\boldsymbol{y}}_1, \ldots, \underline{\boldsymbol{y}}_{k^{(1)}-1}, \underline{\boldsymbol{y}}_{k^{(1)}+1}, \cdots \underline{\boldsymbol{y}}_{2M} \right\}$

$$\underline{\boldsymbol{y}}_{k^{(1)}} = -\frac{1}{\phi_{k^{(1)}}} \sum_{\substack{j=1 \\ j \neq k^{(1)}}}^{2M} \phi_j \underline{\boldsymbol{y}}_j$$

we can eliminate the node $\underline{\boldsymbol{y}}_{k^{(1)}}$ from $A^{(0)}$, without altering the rank of the new coefficient matrix. We set $A^{(1)} = [\underline{\boldsymbol{y}}_1^{(1)}, \underline{\boldsymbol{y}}_2^{(1)}, \ldots, \underline{\boldsymbol{y}}_M^{(1)}] \in \mathbb{R}^{M \times M}$ as defined in step 5), with rank($A^{(0)}$)=rank($A^{(1)}$)=M. Hence, we produce a new probability measure $\tilde{\eta}_{2M-1}$, preserving the $\mathcal{C}oM(\eta_{2M}^{\mu})$.

$$\tilde{\eta}_{2M-1} = \sum_{j=1}^{2M-1} \beta_j^{(1)} \delta_{\underline{\boldsymbol{y}}_j^{(1)}} : \ \left\{ \underline{\boldsymbol{y}}_j^{(1)} \right\}_{j=1}^{2M-1} \subset supp(\eta_{2M}) \ , \ \mathcal{C}oM(\eta_{2M}) = \mathcal{C}oM(\tilde{\eta}_{2M-1})$$

Now, we update $\Phi^{(0)}$, the kernel of $A^{(0)}$, to obtain the kernel of $A^{(1)}$. We can do so, by firstly removing the first null vector $\underline{\boldsymbol{\phi}}_1^{(0)}$ of $\Phi^{(0)} \in \mathbb{R}^{2M \times M}$

$$\Phi^{(0)} \rightarrow \underbrace{\left[\begin{array}{ccc} (\underline{\boldsymbol{\phi}}_1^{(0)})_1 & \cdots & (\underline{\boldsymbol{\phi}}_M^{(0)})_1 \\ \vdots & \cdots & \vdots \\ (\underline{\boldsymbol{\phi}}_1^{(0)})_{2M} & \cdots & (\underline{\boldsymbol{\phi}}_M^{(0)})_{2M} \end{array}\right]}_{M \rightarrow (M-1)} \Bigg\} 2M$$

and subsequently taking the following linear combination of the remaining null vectors of $\Phi^{(0)}$ with the removed null vector $\underline{\boldsymbol{\phi}}_1^{(0)}$, as described in steps 6) and 7)

$$\widehat{\underline{\boldsymbol{\phi}}}_\ell^{(1)} = \underline{\boldsymbol{\phi}}_{\ell+1}^{(0)} - \left[\frac{(\underline{\boldsymbol{\phi}}_{\ell+1}^{(0)})_{k^{(1)}}}{(\underline{\boldsymbol{\phi}}_1^{(0)})_{k^{(1)}}}\right] \underline{\boldsymbol{\phi}}_1^{(0)} \qquad \text{for } \ell \in [\![1, M-1]\!]$$

The above linear combination of null vectors of $\Phi^{(0)}$ introduces a zero in the $k^{(1)}th$ coordinate of each resulting vector $\widehat{\underline{\boldsymbol{\phi}}}_\ell^{(1)} = [(\widehat{\underline{\boldsymbol{\phi}}}_\ell^{(1)})_1, \ldots, (\widehat{\underline{\boldsymbol{\phi}}}_\ell^{(1)})_{2M}]^T \in \mathbb{R}^{2M}$. We proceed by removing the zeroed coordinate in each resulting vector $\widehat{\underline{\boldsymbol{\phi}}}_\ell^{(1)}$

$$\Phi^{(1)} = k^{(1)} \rightarrow \underbrace{\left[\begin{array}{ccc} (\widehat{\underline{\boldsymbol{\phi}}}_1^{(1)})_1 & \cdots & (\widehat{\underline{\boldsymbol{\phi}}}_{M-1}^{(1)})_1 \\ \vdots & \cdots & \vdots \\ 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \\ (\widehat{\underline{\boldsymbol{\phi}}}_1^{(1)})_{2M} & \cdots & (\widehat{\underline{\boldsymbol{\phi}}}_{M-1}^{(1)})_{2M} \end{array}\right]}_{M-1} \Bigg\} 2M \rightarrow \; 2M-1$$

thereby obtaining $\Phi^{(1)} := [\underline{\boldsymbol{\phi}}_1^{(1)}, \underline{\boldsymbol{\phi}}_2^{(1)}, \ldots, \underline{\boldsymbol{\phi}}_{M-1}^{(1)}] \in \mathbb{R}^{(2M-1)\times(M-1)}$ such that $\mathcal{K}er(A^{(1)}) = \mathcal{C}ol(\Phi^{(1)})$, where $\underline{\boldsymbol{\phi}}_i^{(1)}$ for $1 \leq i \leq M-1$ satisfies the expression in step 8).

It is easily verified that $Col(\Phi^{(1)}) \subseteq \mathcal{K}er(A^{(1)})$, as for each $\ell \in [\![1, M-1]\!]$, we have

$$A^{(1)}\underline{\boldsymbol{\phi}}_{\ell}^{(1)} = [\underline{\boldsymbol{y}}_1, \ldots, \underline{\boldsymbol{y}}_{k^{(1)}-1}, \underline{\boldsymbol{y}}_{k^{(1)}+1}, \ldots, \underline{\boldsymbol{y}}_{2M}]\ \underline{\boldsymbol{\phi}}_{\ell}^{(1)} \tag{3.16}$$

$$= \begin{bmatrix} \sum_{\substack{j=1 \\ j\neq k^{(1)}}}^{2M} y_{1,j}(\underline{\boldsymbol{\phi}}_{\ell+1}^{(0)})_j - d_{\ell+1}^{(1)} \sum_{\substack{j=1 \\ j\neq k^{(1)}}}^{2M} y_{1,j}(\underline{\boldsymbol{\phi}}_{1}^{(0)})_j \\ \vdots \\ \sum_{\substack{j=1 \\ j\neq k^{(1)}}}^{2M} y_{M,j}(\underline{\boldsymbol{\phi}}_{\ell+1}^{(0)})_j - d_{\ell+1}^{(1)} \sum_{\substack{j=1 \\ j\neq k^{(1)}}}^{2M} y_{M,j}(\underline{\boldsymbol{\phi}}_{1}^{(0)})_j \end{bmatrix} = \underline{\mathbf{0}}$$

Since for any $\underline{\boldsymbol{\phi}}_{i}^{(0)} \in \mathcal{K}er(A^{(0)})$, we have

$$\begin{aligned} \sum_{\substack{j=1 \\ j\neq k^{(1)}}}^{2M} y_{1,j}(\underline{\boldsymbol{\phi}}_{1}^{(0)})_j &= -y_{1,k^{(1)}}(\underline{\boldsymbol{\phi}}_{i}^{(0)})_{k^{(1)}} \\ &\vdots \\ \sum_{\substack{j=1 \\ j\neq k^{(1)}}}^{2M} y_{M,j}(\underline{\boldsymbol{\phi}}_{1}^{(0)})_j &= -y_{M,k^{(1)}}(\underline{\boldsymbol{\phi}}_{i}^{(0)})_{k^{(1)}} \end{aligned} \tag{3.17}$$

And in particular, by taking $\underline{\boldsymbol{\phi}}_{i}^{(0)} = \underline{\boldsymbol{\phi}}_{1}^{(0)}$ in (3.17) and by substituting (3.17) into (3.16) and using the expression for $d_{\ell+1}^{(1)}$, we obtain the desired equality in (3.16).

Similarly, we can verify $\mathcal{K}er(A^{(1)}) \subseteq \mathcal{C}ol(\Phi^{(1)})$, since $\mathcal{K}er(A^{(0)}) = \mathcal{C}ol(\Phi^{(0)})$, so we have $\underline{\boldsymbol{\phi}}_{m}^{(0)}$, $\underline{\boldsymbol{\phi}}_{i}^{(0)} \in \mathcal{K}er(A^{(0)})$ for any $1 \leq m, i \leq M$ and

$$\underline{\mathbf{0}} = A^{(0)}\underline{\boldsymbol{\phi}}_m^{(0)} = \begin{bmatrix} \sum_{\substack{j=1\\ j\neq k^{(1)}}}^{2M} y_{1,j}(\underline{\boldsymbol{\phi}}_m^{(0)})_j + y_{1,k^{(1)}}\left[\frac{(\underline{\boldsymbol{\phi}}_m^{(0)})_{k^{(1)}}}{(\underline{\boldsymbol{\phi}}_i^{(0)})_{k^{(1)}}}\right](\underline{\boldsymbol{\phi}}_i^{(0)})_{k^{(1)}} \\ \vdots \\ \sum_{\substack{j=1\\ j\neq k^{(1)}}}^{2M} y_{M,j}(\underline{\boldsymbol{\phi}}_m^{(0)})_j + y_{M,k^{(1)}}\left[\frac{(\underline{\boldsymbol{\phi}}_m^{(0)})_{k^{(1)}}}{(\underline{\boldsymbol{\phi}}_i^{(0)})_{k^{(1)}}}\right](\underline{\boldsymbol{\phi}}_i^{(0)})_{k^{(1)}} \\ \\ \end{bmatrix} =$$

$$\begin{bmatrix} \sum_{\substack{j=1\\ j\neq k^{(1)}}}^{2M} y_{1,j}(\underline{\boldsymbol{\phi}}_m^{(0)})_j - \left[\frac{(\underline{\boldsymbol{\phi}}_m^{(0)})_{k^{(1)}}}{(\underline{\boldsymbol{\phi}}_i^{(0)})_{k^{(1)}}}\right]\sum_{\substack{j=1\\ j\neq k^{(1)}}}^{2M} y_{1,j}(\underline{\boldsymbol{\phi}}_i^{(0)})_j \\ \vdots \\ \sum_{\substack{j=1\\ j\neq k^{(1)}}}^{2M} y_{M,j}(\underline{\boldsymbol{\phi}}_m^{(0)})_j - \left[\frac{(\underline{\boldsymbol{\phi}}_m^{(0)})_{k^{(1)}}}{(\underline{\boldsymbol{\phi}}_i^{(0)})_{k^{(1)}}}\right]\sum_{\substack{j=1\\ j\neq k^{(1)}}}^{2M} y_{M,j}(\underline{\boldsymbol{\phi}}_i^{(0)})_j \\ \\ \end{bmatrix} =$$

$$\begin{bmatrix} \sum_{\substack{j=1\\ j\neq k^{(1)}}}^{2M} y_{1,j}\left((\underline{\boldsymbol{\phi}}_m^{(0)})_j - \left[\frac{(\underline{\boldsymbol{\phi}}_m^{(0)})_{k^{(1)}}}{(\underline{\boldsymbol{\phi}}_i^{(0)})_{k^{(1)}}}\right](\underline{\boldsymbol{\phi}}_i^{(0)})_j\right) \\ \vdots \\ \sum_{\substack{j=1\\ j\neq k^{(1)}}}^{2M} y_{M,j}\left((\underline{\boldsymbol{\phi}}_m^{(0)})_j - \left[\frac{(\underline{\boldsymbol{\phi}}_m^{(0)})_{k^{(1)}}}{(\underline{\boldsymbol{\phi}}_i^{(0)})_{k^{(1)}}}\right](\underline{\boldsymbol{\phi}}_i^{(0)})_j\right) \\ \\ \end{bmatrix} = A^{(1)}\underline{\boldsymbol{\phi}}_\ell^{(1)}$$

where the first equality holds by (3.17), since $\underline{\boldsymbol{\phi}}_i^{(0)} \in \mathcal{K}er(A^{(0)})$. Hence, we conclude that any $\underline{\boldsymbol{\phi}}_\ell^{(1)} = [(\underline{\boldsymbol{\phi}}_\ell^{(1)})_1,\ (\underline{\boldsymbol{\phi}}_\ell^{(1)})_2, \ldots, (\underline{\boldsymbol{\phi}}_\ell^{(1)})_{2M-1}]^T \in \mathcal{K}er(A^{(1)})$ must be of the form

$$(\underline{\boldsymbol{\phi}}_\ell^{(i)})_j := \begin{cases} \left((\underline{\boldsymbol{\phi}}_m^{(0)})_j - \left[\frac{(\underline{\boldsymbol{\phi}}_m^{(0)})_{k^{(1)}}}{(\underline{\boldsymbol{\phi}}_i^{(0)})_{k^{(1)}}}\right](\underline{\boldsymbol{\phi}}_i^{(0)})_j\right) & \text{for } j \in [\![1, k^{(1)}-1]\!] \\ \left((\underline{\boldsymbol{\phi}}_m^{(0)})_{j+1} - \left[\frac{(\underline{\boldsymbol{\phi}}_m^{(0)})_{k^{(1)}}}{(\underline{\boldsymbol{\phi}}_i^{(0)})_{k^{(1)}}}\right](\underline{\boldsymbol{\phi}}_i^{(0)})_{j+1}\right) & \text{for } j \in [\![k^{(1)}, 2M-1]\!] \end{cases}$$

and hence by choosing $m = \ell + 1$ and $i = 1$, we obtain the expression in step 8) and $\underline{\boldsymbol{\phi}}_\ell^{(1)} \in \mathcal{C}ol(\Phi^{(1)})$.

At the end of iteration $(i-1) \leq M-1$, we have produced:

$$A^{(i-1)} = [\underline{\boldsymbol{y}}_1^{(i-1)}, \underline{\boldsymbol{y}}_2^{(i-1)}, \ldots, \underline{\boldsymbol{y}}_{2M-(i-1)}^{(i-1)}] \in \mathbb{R}^{M \times (2M-(i-1))},\ \text{rank}(A^{(i-1)}) = M$$

$$\underline{\boldsymbol{\beta}}^{(i-1)} = [\beta_1^{(i-1)}, \beta_2^{(i-1)}, \ldots, \beta_{2M-(i-1)}^{(i-1)}]^T \in \mathbb{R}_+^{2M-(i-1)},\ \sum_{j=1}^{2M-(i-1)} \beta_j^{(i-1)} = 1$$

$$\Phi^{(i-1)} = [\underline{\boldsymbol{\phi}}_1^{(i-1)}, \underline{\boldsymbol{\phi}}_2^{(i-1)}, \ldots, \underline{\boldsymbol{\phi}}_{M-(i-1)}^{(i-1)}] \in \mathbb{R}^{(2M-(i-1)) \times (M-(i-1))}\ ,\ Col(\Phi^{(i-1)}) = \mathcal{K}er(A^{(i-1)})$$

$$\tilde{\eta}_{2M-(i-1)} = \sum_{j=1}^{2M-(i-1)} \beta_j^{(i-1)} \delta_{\underline{\boldsymbol{y}}_j^{(i-1)}} :\ \{\underline{\boldsymbol{y}}_j^{(i-1)}\}_{j=1}^{2M-(i-1)} \subset supp(\eta_{2M})\ ,$$

$$\mathcal{C}oM(\eta_{2M}) = \mathcal{C}oM(\tilde{\eta}_{2M-(i-1)})$$

Hence, on iteration i at step 3) we select $\underline{\boldsymbol{\phi}}_1^{(i-1)} \in \Phi^{(i-1)}$ and since at least one $(\underline{\boldsymbol{\phi}}_1^{(i-1)})_j > 0$, we have

$$\mathcal{C}oM(\tilde{\eta}_{2M-(i-1)}) = \sum_{j=1}^{2M-(i-1)} \beta_j^{(i-1)} \underline{\boldsymbol{y}}_j^{(i-1)} = \sum_{j=1}^{2M-(i-1)} \beta_j^{(i-1)} \underline{\boldsymbol{y}}_j^{(i-1)}$$

$$- \alpha_{(i)} \left(\sum_{j=1}^{2M-(i-1)} (\underline{\boldsymbol{\phi}}_1^{(i-1)})_j \underline{\boldsymbol{y}}_j^{(i-1)} \right) = \sum_{j=1}^{2M-(i-1)} \left(\beta_j^{(i-1)} - \alpha_{(i)} (\underline{\boldsymbol{\phi}}_1^{(i-1)})_j \right) \underline{\boldsymbol{y}}_j^{(i-1)}$$

$$= \sum_{j=1}^{2M-i} \beta_j^{(i)} \underline{\boldsymbol{y}}_j^{(i)} = \mathcal{C}oM(\tilde{\eta}_{2M-i})$$

where $\alpha_{(i)}$ is as defined in step 3), $\beta_j^{(i)}$ is as defined in step 4) and $\underline{\boldsymbol{y}}_j^{(i)}$ is as defined in step 5).

Now, we update $\Phi^{(i-1)}$, the kernel of $A^{(i-1)}$, to obtain the kernel of $A^{(i)}$. We can do this by firstly removing the first null vector $\underline{\boldsymbol{\phi}}_1^{(i-1)}$ of $\Phi^{(i-1)} \in \mathbb{R}^{2M-(i-1) \times M-(i-1)}$

$$\Phi^{(i-1)} \rightarrow \underbrace{\begin{bmatrix} (\underline{\boldsymbol{\phi}}_1^{(i-1)})_1 & \cdots & (\underline{\boldsymbol{\phi}}_{M-(i-1)}^{(i-1)})_1 \\ \vdots & \cdots & \vdots \\ \vdots & \ddots & \vdots \\ (\underline{\boldsymbol{\phi}}_1^{(i-1)})_{2M-(i-1)} & \cdots & (\underline{\boldsymbol{\phi}}_{M-(i-1)}^{(i-1)})_{2M-(i-1)} \end{bmatrix}}_{M-(i-1) \rightarrow M-i} \Bigg\} \begin{matrix} \\ \\ \\ \\ 2M\ -(i-1) \end{matrix}$$

and taking the following linear combination of the remaining null vectors of $\Phi^{(i-1)}$ with the removed null vector $\underline{\boldsymbol{\phi}}_1^{(i-1)}$, as described in steps 6) and 7)

$$\widehat{\underline{\boldsymbol{\phi}}}_\ell^{(i)} = \underline{\boldsymbol{\phi}}_{\ell+1}^{(i-1)} - \left[\frac{(\underline{\boldsymbol{\phi}}_{\ell+1}^{(i-1)})_{k^{(i)}}}{(\underline{\boldsymbol{\phi}}_1^{(i-1)})_{k^{(i)}}}\right]\underline{\boldsymbol{\phi}}_1^{(i-1)} \qquad \text{for } \ell \in [\![1, M-i]\!]$$

The above linear combination of null vectors of $\Phi^{(i-1)}$ introduces a zero in the $k^{(i)}th$ coordinate of each resulting vector $\widehat{\underline{\boldsymbol{\phi}}}_\ell^{(i)} = [(\widehat{\underline{\boldsymbol{\phi}}}_\ell^{(i)})_1, \ldots, (\widehat{\underline{\boldsymbol{\phi}}}_\ell^{(i)})_{2M-(i-1)}]^T \in \mathbb{R}^{(2M-(i-1))}$. We proceed by removing the zeroed coordinate in each resulting vector $\widehat{\underline{\boldsymbol{\phi}}}_\ell^{(i)}$

$$\Phi^{(i)} = k^{(i)} \rightarrow \underbrace{\left[\begin{array}{ccc} (\widehat{\underline{\boldsymbol{\phi}}}_1^{(i)})_1 & \cdots & (\widehat{\underline{\boldsymbol{\phi}}}_{M-i}^{(i)})_1 \\ \vdots & \cdots & \vdots \\ 0 & & 0 \\ \vdots & \ddots & \vdots \\ (\widehat{\underline{\boldsymbol{\phi}}}_1^{(i)})_{2M-(i-1)} & \cdots & (\widehat{\underline{\boldsymbol{\phi}}}_{M-i}^{(i)})_{2M-(i-1)} \end{array}\right]}_{M-i} \Bigg\} \; 2M-(i-1) \rightarrow 2M-i$$

so as to obtain $\Psi^{(i)} := [\underline{\boldsymbol{\phi}}_1^{(i)}, \underline{\boldsymbol{\phi}}_2^{(i)}, \ldots, \underline{\boldsymbol{\phi}}_{M-i}^{(i)}] \in \mathbb{R}^{(2M-i)\times(M-i)}$ such that $\mathcal{K}er(A^{(i)}) = \mathcal{C}ol(\Phi^{(i)})$, where $\underline{\boldsymbol{\phi}}_m^{(i)}$ for $1 \leq m \leq M-1$ satisfies the expression in step 8). It is easily verified that $Col(\Phi^{(i)}) \subseteq \mathcal{K}er(A^{(i)})$, for each $\ell \in [\![1, M-i]\!]$, we have

$$A^{(i)}\underline{\boldsymbol{\phi}}_\ell^{(i)} = \left[\begin{array}{c} \sum_{j=1,\ j\neq k^{(1)},\ldots,k^{(i)}}^{2M} y_{1,j}(\underline{\boldsymbol{\phi}}_{\ell+1}^{(i-1)})_j - d_{\ell+1}^{(i)} \sum_{j=1,\ j\neq k^{(1)},\ldots,k^{(i)}}^{2M} y_{1,j}(\underline{\boldsymbol{\phi}}_1^{(i-1)})_j \\ \vdots \\ \sum_{j=1,\ j\neq k^{(1)},\ldots,k^{(i)}}^{2M} y_{M,j}(\underline{\boldsymbol{\phi}}_{\ell+1}^{(i-1)})_j - d_{\ell+1}^{(i)} \sum_{j=1,\ j\neq k^{(1)},\ldots,k^{(i)}}^{2M} y_{M,j}(\underline{\boldsymbol{\phi}}_1^{(i-1)})_j \end{array}\right] = \underline{\mathbf{0}} \tag{3.18}$$

Since for $\underline{\boldsymbol{\phi}}_1^{(0)} \in \mathcal{K}er(A^{(i-1)})$, we have

$$\begin{array}{c} \sum_{j=1,\ j\neq k^{(1)},\ldots,k^{(i)}}^{2M} y_{1,j}(\underline{\boldsymbol{\phi}}_1^{(0)})_j = -y_{1,k^{(i)}}(\underline{\boldsymbol{\phi}}_1^{(0)})_{k^{(1)}} \\ \vdots \\ \sum_{j=1,\ j\neq k^{(1)},\ldots,k^{(i)}}^{2M} y_{M,j}(\underline{\boldsymbol{\phi}}_1^{(0)})_j = -y_{M,k^{(i)}}(\underline{\boldsymbol{\phi}}_1^{(0)})_{k^{(1)}} \end{array}$$

By substituting the above set of equations into (3.18) and using the expression for

$d^{(i)}_{\ell+1}$, we obtain the desired equality in (3.18). Similarly, it is easily verified that $\mathcal{K}er(A^{(i)}) \subseteq Col(\Phi^{(i)})$.

Hence, at the end of iteration i, we have produced:

$$
\begin{aligned}
&A^{(i)} = [\underline{\boldsymbol{y}}^{(i)}_1, \underline{\boldsymbol{y}}^{(i)}_2, \ldots, \underline{\boldsymbol{y}}^{(i)}_{2M-i}] \in \mathbb{R}^{M\times(2M-i)},\ \mathrm{rank}(A^{(i)}) = M \\
&\underline{\boldsymbol{\beta}}^{(i)} = [\beta^{(i)}_1, \beta^{(i)}_2, \ldots, \beta^{(i)}_{2M-i}]^T \in \mathbb{R}^{2M-i}_+,\ \sum_{j=1}^{2M-i} \beta^{(i)}_j = 1 \\
&\Phi^{(i)} = [\underline{\boldsymbol{\phi}}^{(i)}_1, \underline{\boldsymbol{\phi}}^{(i)}_2, \ldots, \underline{\boldsymbol{\phi}}^{(i)}_{M-i}] \in \mathbb{R}^{(2M-i)\times(M-i)}\ ,\ Col(\Phi^{(i)}) = \mathcal{K}er(A^{(i)}) \\
&\tilde{\eta}_{2M-i} = \sum_{j=1}^{2M-i} \beta^{(i)}_j \delta_{\underline{\boldsymbol{y}}^{(i)}_j} :\ \{\underline{\boldsymbol{y}}^{(i)}_j\}_{j=1}^{2M-i} \subset supp(\eta_{2M})\ , \\
&\mathcal{C}oM(\eta_{2M}) = \mathcal{C}oM(\tilde{\eta}_{2M-i})
\end{aligned}
$$

and whence at the end of iteration $i = M$, we have produced the reduced measure $\tilde{\eta}_M$ for η_{2M}

$$
\begin{aligned}
&\tilde{\eta}_M = \sum_{k=1}^{M} \beta^{(M)}_k \delta_{\underline{\boldsymbol{y}}^{(M)}_k} = \sum_{k=1}^{M} \hat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}} :\ \{\underline{\boldsymbol{y}}_{i_k}\}_{k=1}^{M} \subset supp(\eta_{2M})\ , \\
&\sum_{k=1}^{M} \hat{\beta}_k = 1,\ \mathcal{C}oM(\eta_{2M}) = \mathcal{C}oM(\tilde{\eta}_M)
\end{aligned}
$$

□

Remark 3.3.1. *Theoretical Complexity of M-Tree Measure Reduction Algorithm*

Let $C(\eta_{2M}, \tilde{\eta}_M)$ denote the complexity to perform an M-Tree Measure Reduction. Then $C(\eta_{2M}, \tilde{\eta}_M)$ is dominated by two operations: the first is the computation of the $\mathcal{K}er(A^{(0)})$, which is a $M \times 2M$ matrix in step 2), which by the standard "Golub-Kahan-Reinsch" SVD algorithm see Chapter 8.6.4 [106], takes

$$4M^2 \times (2M) + 8M \times (2M)^2 + 9 \times (2M)^3 = 112M^3 + \textit{lower order terms}$$

and secondly updating the kernel $\Phi^{(i-1)} \to \Phi^{(i)}$ in steps 6)-8) at each iteration $i = 1 : M$. Given $\Phi^{(i-1)} = [\underline{\boldsymbol{\phi}}_1^{(i-1)}, \underline{\boldsymbol{\phi}}_2^{(i-1)}, \ldots, \underline{\boldsymbol{\phi}}_{M-(i-1)}^{(i-1)}] \in \mathbb{R}^{(2M-(i-1))\times(M-(i-1))}$, the latter operation requires the deletion of the first column $\underline{\boldsymbol{\phi}}_1^{(i-1)}$ from $\Phi^{(i-1)}$ and the subsequent update of each of the remaining columns of $\Phi^{(i-1)}$

$$\underline{\boldsymbol{\phi}}_{\ell+1}^{(i-1)} - \left[\frac{(\underline{\boldsymbol{\phi}}_{\ell+1}^{(i-1)})_{k^{(i)}}}{(\underline{\boldsymbol{\phi}}_1^{(i-1)})_{k^{(i)}}}\right] \underline{\boldsymbol{\phi}}_1^{(i-1)} \qquad \textit{for } \ell \in [\![1, M-i]\!]$$

which is equivalent to subtracting two matrices of dimensions $(2M-(i-1)) \times (M-i)$ and the subsequent deletion of row $k^{(i)}$ in the resulting matrix. Since the kernel update is performed for every iteration $i = 1 : M$, we have a total computation of order

$$M \times \big(2M - (i-1)\big) \times \big(M - i\big) = 2M^3 + \textit{lower order terms}$$

Hence, clearly $C(\eta_{2M}, \tilde{\eta}_M) = O\big(M^3\big)$

Algorithm 7: Recombination Algorithm with M-Tree Measure Reduction

Input: Let $\nu = \sum_{i=1}^{N} \omega_i \delta_{\underline{\boldsymbol{x}}_i}$ be an atomic probability measure on $\mathbb{R}^d$, $supp(\nu) \subseteq K$, $\Psi = (\psi_1, \ldots, \psi_M) : \ K \to \mathbb{R}^M$ be a continuos, ν-integrable map

1) Produce the push-forward of ν through Ψ

$$(\Psi_* \nu) = \sum_{i=1}^{N} \omega_i \delta_{\Psi(\underline{\boldsymbol{x}}_i)}$$

2) Produce, via RHFC Algorithm, $2M$-Tree form of $(\Psi_* \nu)$

$$\eta_{2M}^0 = \sum_{i=1}^{2M} \beta_i^0 \delta_{\underline{\boldsymbol{y}}_i^0} \qquad \beta_i^0 = P(T_i^0), \ \underline{\boldsymbol{y}}_i^0 = R(T_i^0)$$

and set $j = 0$

3) Produce, via $2M$-Tree Measure Reduction Algorithm and tree updating, a reduced tree measure η_M^j for η_{2M}^j

$$\eta_M^j = \sum_{i=1}^{M} \widehat{\beta}_k^j \delta_{\underline{\boldsymbol{y}}_{i_k}^j} \qquad \widehat{\beta}_k^j = P(\widehat{T}_{i_k}^j), \ \underline{\boldsymbol{y}}_{i_k}^j = R(\widehat{T}_{i_k}^j)$$

where each tree $\widehat{T}_{i_k}^j$ is a weight-updated tree $T_{i_k}^j$, corresponding to the root weight update $\beta_k^j \to \widehat{\beta}_k^j$

If $\text{depth}(\widehat{T}_{i_k}^j) > 1$, for some $k \in [\![1, M]\!]$, proceed to step 4), else go to step 6)

4) Split each binary tree $\widehat{T}_{i_k}^j \in \{\widehat{T}_{i_1}^j, \ldots, \widehat{T}_{i_M}^j\}$ into two subtrees, by removing its root node

$$\Big(P(\widehat{T}_{i_k}^j), \ R(\widehat{T}_{i_k}^j)\Big)$$

$$\Big(P(\tilde{T}_{i_{2k-1}}^j), \ R(\tilde{T}_{i_{2k-1}}^j)\Big) \quad \Big(P(\tilde{T}_{i_{2k}}^j), \ R(\tilde{T}_{i_{2k}}^j)\Big)$$

5) Set $\left\{T_1^{j+1},\ldots,T_{2M}^{j+1}\right\} := \left\{\tilde{T}_{i_1}^{j},\ldots,\tilde{T}_{i_{2M}}^{j}\right\}$ and produce a $2M$-Tree form of $(\Psi_*\nu)$, given by

$$\eta_{2M}^{j+1} = \sum_{i=1}^{2M} \beta_i^{j+1}\delta_{\underline{\boldsymbol{y}}_i^{j+1}} \qquad \beta_i^{j+1} = P(T_i^{j+1}),\ \underline{\boldsymbol{y}}_i^{j+1} = R(T_i^{j+1})$$

set $j = j+1$ and go to step 3).

6) Denote the output measure from step 3) by

$$\widehat{(\Psi_*\nu)} = \sum_{k=1}^{M} \widehat{\omega}_k \delta_{\Psi(\underline{\boldsymbol{x}}_{i_k})}$$

7) **Output**: Ψ-generalised Carathéodory cubature measure for ν:

$$\widehat{\nu}_\Psi = \sum_{k=1}^{M} \widehat{\omega}_k \delta_{\underline{\boldsymbol{x}}_{i_k}}$$

Remark 3.3.2. *In the preceding algorithm: the Recombination Algorithm with M-Tree Measure Reduction, we have assumed that each measure η_{2M}^{j}, generating a coefficient matrix $A_j^{(0)} := [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_{2M}]$ has rank($A_j^{(0)}$)=$M' = M$, as described in the M-Tree Measure Reduction, for simplicity of exposition.*

3.3.0.13 Theoretical Complexity of the Recombination Algorithm with M-Tree Measure Reduction

In step 1), we produce the push-forward of $\nu = \sum_{i=1}^{N} \omega_i \delta_{\underline{\boldsymbol{x}}_i}$ with $supp(\nu) \subseteq \mathbb{R}^d$ through the M-dimensional map $\Psi = (\psi_1, \ldots, \psi_M)$, yielding

$$(\Psi_*\nu) = \sum_{i=1}^{N} \omega_i \delta_{\Psi(\underline{\boldsymbol{x}}_i)} \text{ with } supp(\Psi_*\nu) \subseteq \mathbb{R}^M$$

which by construction has a centre of mass, satisfying

$$\begin{aligned}\mathcal{C}oM\big(\Psi_*\nu\big) &= \Big[\sum_{i=1}^{N}\omega_i\psi_1(\underline{\boldsymbol{x}}_i),\ldots,\sum_{i=1}^{N}\omega_i\psi_M(\underline{\boldsymbol{x}}_i)\Big]^T\\ &= \Big[\int_K \psi_1(\underline{\boldsymbol{x}})d\nu,\ldots,\int_K \psi_M(\underline{\boldsymbol{x}})d\nu\Big]^T\end{aligned}$$

Thus, we have transformed out original goal of constructing a Ψ-generalised Carathéodory cubature measure for ν: $\widehat{\nu}_\Psi = \sum_{k=1}^{M}\widehat{\omega}_k\delta_{\underline{\boldsymbol{x}}_{i_k}}$, that is a measure on $\mathbb{R}^d$ satisfying

$$\begin{aligned}&1)\ \int_{\mathbb{R}^d}\psi_j(\underline{\boldsymbol{x}})d\nu(\underline{\boldsymbol{x}}) = \int_{\mathbb{R}^d}\psi_j(\underline{\boldsymbol{x}})d\widehat{\nu}_\Psi(\underline{\boldsymbol{x}}) \quad \forall j\in[\![1,M]\!] \qquad (3.19)\\ &2)\ supp(\widehat{\nu}_\Psi)\subseteq supp(\nu)\\ &3)\ card\big(supp(\widehat{\nu}^{\mu}_\Psi)\big)\leq M+1\end{aligned}$$

into constructing a reduced measure $(\widehat{\Psi_*\nu}) = \sum_{k=1}^{M}\widehat{\omega}_k\delta_{\Psi(\underline{\boldsymbol{x}}_{i_k})}$ for $(\Psi_*\nu)$, on $\mathbb{R}^M$, satisfying

$$\begin{aligned}&1)\ \mathcal{C}oM(\Psi_*\nu) = \sum_{k=1}^{N}\omega_i\Psi(\underline{\boldsymbol{x}}_i) = \sum_{i=1}^{M}\widehat{\omega}_k\Psi(\underline{\boldsymbol{x}}_{i_k}) = \mathcal{C}oM(\widehat{\Psi_*\nu})\\ &2)\ supp(\widehat{\Psi_*\nu})\subseteq supp(\Psi_*\nu)\\ &3)\ card(supp(\widehat{\Psi_*\nu}))\leq M+1\end{aligned}$$

In step 1) the cost of producing the push-forward of ν through the map $\Psi := (\psi_1,\ldots,\psi_M)$ is dMN, since we are evaluating an M-dimensional map at N d-dimensional points.

In step 2) we proceed to cluster the measure $(\Psi_*\nu)$, via RHFC Algorithm. For simplicity of exposition of step 2), we assume that $N = 2^S$ and $2M = 2^R$ for some $S > R$. Then, η^0_{2M}, produced in step 2), is given by

$$\eta^0_{2M} = \sum_{i=1}^{2M}\beta^0_i\delta_{\underline{\boldsymbol{y}}^0_i}: \quad \beta^0_i = P(T^0_i), \quad \underline{\boldsymbol{y}}^0_i = R(T^0_i) \quad \text{for} \quad 1\leq i\leq 2M$$

where each tree in the binary forest $\big\{T^0_1,\ldots,T^0_{2M}\big\}$, has $depth(T^0_i) = \lceil log_2\big(\frac{2^S}{2^R}\big)\rceil = (S-R)$ and an associated collection of leaf nodes of cardinality $card(l(T^0_i)) = 2^{S-R}$.

On the first pass through the Recombination Algorithm, the M-Tree measure reduction in step 3) eliminates M trees of depth $(S-R)$, thereby eliminating $M \cdot 2^{(S-R)}$ points from $supp(\nu)$, the support of the input measure. Clearly, on the ith pass through the Recombination Algorithm, M-Tree Measure Reduction eliminates all trees of depth $(S-R)-(i-1)$ and hence $M \cdot 2^{(S-R)-(i-1)} = 2^S/2^i$ points from the $supp(\nu)$, this halves the number of points, $2^S/2^{(i-1)}$, left in the $supp(\nu)$ from the $(i-1)$th pass through the algorithm.

Hence, the Recombination Algorithm with M-Tree Measure Reduction requires a total of $log(N/2M) - 1 = (S-R) - 1$ iterations through steps 3)-5). When N, $2M$ are not exact powers of two, the number of iterations through steps 3)-5) of the Recombination Algorithm is of order $O\big(log(N/2M)\big)$.

Now, let $C(\eta_{2M}^j, \tilde{\eta}_M^j)$ denote the complexity of performing an M-Tree Measure Reduction, let $C_{up}(\tilde{\eta}_M^j, \eta_M^j)$ denote the complexity of updating the weights down each tree $T_{i_k} \in \{T_{i_1}, \ldots, T_{i_M}\}$ in the $supp(\eta_M^j)$ and let $C_{spl}(\tilde{\eta}_M^j, \eta_{2M}^{j+1})$ denote the complexity of splitting each tree in the $supp(\eta_M^j)$ to produce the $supp(\eta_{2M}^{j+1})$.

Then, the computational complexity of the Recombination Algorithm is given by

$$O\Big(NM + log\big(N/(2M)\big)\big[C(\eta_{2M}^j, \tilde{\eta}_M^j) + C_{up}(\tilde{\eta}_M^j, \eta_M^j) + C_{spl}(\tilde{\eta}_M^j, \eta_{2M}^{j+1})\big]\Big) \tag{3.20}$$

where the first term arises from the construction of the push-forward of ν through Ψ in step 1) and RHFC Algorithm in step 2) and the second term arises from the iteration through steps 3)-5). We already know that, $C(\eta_{2M}^j, \tilde{\eta}_M^j) = O(M^3)$ and as stated in Section 3.1.0.7, we can produce a weight-updated tree $\widehat{T}_{i_k}$ from T_{i_k} in $O(1)$, by employing trivial parallelisation. Therefore we have $C_{up}(\tilde{\eta}_M^j, \eta_M^j) = O(M)$. In steps 4) and 5) the tree splitting procedure and thus the construction of η_{2M}^{j+1} from η_M^j, requires referencing the left and right subtrees of each binary tree in η_M^j and thus can be accomplished in order $C_{spl}(\tilde{\eta}_M^j, \eta_{2M}^{j+1}) = O(M)$.

Hence, the cost of the Recombination Algorithm with M-Tree Measure Reduction is given by

$$O\Big(NM + log(N/2M)M^3\Big)$$

3.3.1 Degenerate Measure Reduction Algorithms and Practical Implementation Issues

Two types of degeneracies can occur for which the Measure Reduction Algorithms, discussed in the previous section, have to be modified.

The first type of degeneracy occurs when given an atomic probability measure

$$\eta_L = \sum_{i=1}^{L} \beta_i \underline{\boldsymbol{y}}_i$$

resulting in a coefficient matrix $A^{(0)} := [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_L]$, with $rank(A^{(0)}) = M^{'} < M$. Under this scenario, the reduced measure $\tilde{\eta}_{M'}$ will contain $M^{'} < M$ nodes. We have already addressed this type of degeneracy for $L = M + 1$, so in the following we shall address this type of degeneracy for $L = 2M$.

The second type of degeneracy occurs when the coefficient matrix $A^{(0)} := [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_L]$ has $rank(A^{(0)}) = M$, whilst the reduced measure $\tilde{\eta}_{M'}$ contains $M^{'} < M$ nodes. This scenario occurs when $\mathcal{C}oM(\eta_L)$ lies on one of the faces of the Convex Hull of the support of the reduced tree measure $\tilde{\eta}_{M'}$.

Example 3.3.3. *Suppose we have four points on a line in* $[0,1]^2$ *with uniform weights*

$$\nu = \sum_{i=1}^{4} \omega_i \delta_{\underline{\boldsymbol{x}}_i}, \qquad \textit{where} \quad \omega_i = \frac{1}{4}, \quad \textit{for} \quad i \in [\![1,4]\!] \quad \textit{and}$$
$$\underline{\boldsymbol{x}}_1 = [0,1]^T, \quad \underline{\boldsymbol{x}}_2 = [1,0]^T, \quad \underline{\boldsymbol{x}}_3 = [0.75, 0.25]^T, \quad \underline{\boldsymbol{x}}_4 = [0.25, 0.75]^T$$

and let $\Psi = (\psi_1, \psi_2, \psi_3) : \ [0,1]^2 \to \mathbb{R}^3$, *be given* $\Psi(\underline{\boldsymbol{x}}) = \Psi(x^1, x^2) = [1, x^1, x^2]^T$. *Then the push-forward of* ν *through* Ψ *is given by* $(\Psi_* \nu) = \sum_{i=1}^{4} \omega_i \delta_{\Psi(\underline{\boldsymbol{x}}_i)}$ *and we set*

$$A^{(0)} = [\Psi(\underline{\boldsymbol{x}}_1), \Psi(\underline{\boldsymbol{x}}_2), \Psi(\underline{\boldsymbol{x}}_3), \Psi(\underline{\boldsymbol{x}}_4)] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0.75 & 0.25 \\ 1 & 0 & 0.25 & 0.75 \end{bmatrix}$$

$$\mathcal{K}er(A^{(0)}) = span\{\underline{\boldsymbol{\lambda}}_1, \underline{\boldsymbol{\lambda}}_2\} = span\{[-0.25, -0.75, 1, 0]^T, [-0.75, -0.25, 0, 1]^T\}$$

Since $rank(A^{(0)}) = 2 < 3$, $dim\big(\mathcal{K}er(A^{(0)})\big) = 2$ *and* $card(\Psi_*\nu) = 4$, *we set* $\eta_{2M} = (\Psi_*\nu)$ *and proceed with* M*-Tree Measure Reduction, with* $M = 2$. *Setting*

$$\Phi^{(0)}_{(1)} = [\underline{\boldsymbol{\lambda}}_1, \underline{\boldsymbol{\lambda}}_2], \quad \Phi^{(0)}_{(2)} = [-\underline{\boldsymbol{\lambda}}_1, -\underline{\boldsymbol{\lambda}}_2], \quad \Phi^{(0)}_{(3)} = [\underline{\boldsymbol{\lambda}}_2, -\underline{\boldsymbol{\lambda}}_1], \quad \Phi^{(0)}_{(4)} = [-\underline{\boldsymbol{\lambda}}_2, \underline{\boldsymbol{\lambda}}_1]$$

for each kernel $\Phi^{(0)}_{(j)}$ *we obtain the corresponding* Ψ*-generalised Carathéodory cubature measure*

$$\nu^{(1)}_{\psi} = \sum_{i=1}^{2} \widehat{\omega}^{(1)}_i \delta_{\underline{\widehat{\boldsymbol{x}}}^{(1)}_i}, \quad \widehat{\omega}^{(1)}_1 = \widehat{\omega}^{(1)}_2 = \frac{1}{2}, \quad \underline{\widehat{\boldsymbol{x}}}^{(1)}_1 = [0,1]^T, \quad \underline{\widehat{\boldsymbol{x}}}^{(1)}_2 = [1,0]^T$$

$$\nu^{(2)}_{\psi} = \sum_{i=1}^{2} \widehat{\omega}^{(2)}_i \delta_{\underline{\widehat{\boldsymbol{x}}}^{(2)}_i}, \quad \widehat{\omega}^{(2)}_1 = \widehat{\omega}^{(2)}_2 = \frac{1}{2}, \quad \underline{\widehat{\boldsymbol{x}}}^{(2)}_1 = [0.75, 0.25]^T, \quad \underline{\widehat{\boldsymbol{x}}}^{(2)}_2 = [0.25, 0.75]^T$$

$$\nu^{(3)}_{\psi} = \sum_{i=1}^{2} \widehat{\omega}^{(3)}_i \delta_{\underline{\widehat{\boldsymbol{x}}}^{(3)}_i}, \quad \widehat{\omega}^{(3)}_1 = \widehat{\omega}^{(3)}_2 = \frac{1}{2}, \quad \underline{\widehat{\boldsymbol{x}}}^{(3)}_1 = [0,1]^T, \quad \underline{\widehat{\boldsymbol{x}}}^{(3)}_2 = [0.75, 0.25]^T$$

$$\nu^{(4)}_{\psi} = \sum_{i=1}^{2} \widehat{\omega}^{(1)}_i \delta_{\underline{\widehat{\boldsymbol{x}}}^{(4)}_i}, \quad \widehat{\omega}^{(4)}_1 = \widehat{\omega}^{(4)}_2 = \frac{1}{2}, \quad \underline{\widehat{\boldsymbol{x}}}^{(4)}_1 = [1,0]^T, \quad \underline{\widehat{\boldsymbol{x}}}^{(4)}_2 = [0.25, 0.75]^T$$

In the aforementioned example the degeneracy arises as the result of the underlying coefficient matrix being rank-deficient, $rank(A^{(0)}) = 2 = M' < M = 3$. This occurs because the original cubature nodes are collinear. Similarly for higher dimensions, this degeneracy arises when the nodes are contained in an affine subspace $S \subset \mathbb{R}^M$, with $dim(S) < M$.

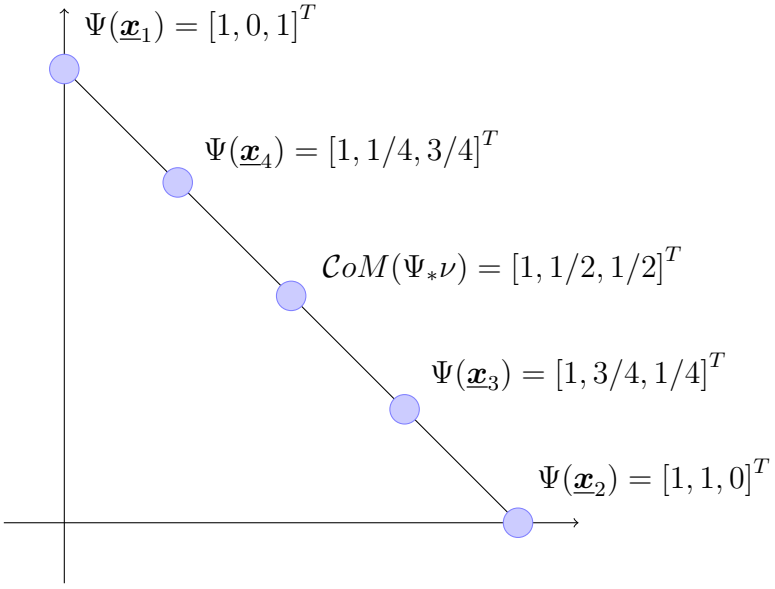


Figure 3.1: Rank-Deficient Degeneracy

To deal with this type of degeneracy, we need to modify the M-Tree Measure Reduction Algorithm.

Let $\eta_{2M} = \sum_{i=1}^{2M} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$ be given, set $A^{(0)} := [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_{2M}]$ and assume that has $rank(A^{(0)}) = M^{'} < M$. In order to construct the reduced measure $\tilde{\eta}_{M'}$ for η_{2M}, we need perform the $(2M - M^{'})$-Tree Measure Reduction. To accomplish this we proceed as in step 2) of the M-Tree Measure Reduction and we construct the basis for the kernel of $A^{(0)}$

$$\Phi^{(0)} := [\underline{\boldsymbol{\phi}}_1^{(0)}, \underline{\boldsymbol{\phi}}_2^{(0)}, \ldots, \underline{\boldsymbol{\phi}}_{2M-M'}^{(0)}] \in \mathbb{R}^{2M \times (2M-M^{'})}$$

where $\mathcal{C}ol(\Phi^{(0)}) = \mathcal{K}er(A^{(0)})$, since $dim(\mathcal{K}er(A^{(0)})) = 2M - M^{'} > M$. Since there are $(2M - M^{'})$ null vectors of $A^{(0)}$, we iterate through steps $3) - 5)$, $(2M - M^{'})$ times, instead of M, producing in steps $9) - 10)$ with the reduced weight vector and coefficient matrix

$$\underline{\hat{\boldsymbol{\beta}}} = [\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_{M'}]^T := [\beta_1^{(2M-M^{'})}, \beta_2^{(2M-M^{'})}, \ldots, \beta_{M'}^{(2M-M^{'})}]^T = \underline{\boldsymbol{\beta}}^{(2M-M^{'})}$$

$$\hat{A} = [\underline{\boldsymbol{y}}_{i_1}, \underline{\boldsymbol{y}}_{i_2}, \ldots, \underline{\boldsymbol{y}}_{i_{M'}}] := [\underline{\boldsymbol{y}}_1^{(2M-M^{'})}, \underline{\boldsymbol{y}}_2^{(2M-M^{'})}, \ldots, \underline{\boldsymbol{y}}_{M'}^{(2M-M^{'})}] = A^{(2M-M^{'})}$$

and hence we obtain $\tilde{\eta}_{M'}$ the reduced tree measure for η_{2M}, given by

$$\tilde{\eta}_{M'} = \sum_{k=1}^{M^{'}} \hat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}}$$

The next example deals with the second type of degeneracy.

Example 3.3.4. *Suppose we have a uniformly weighted square in* $\mathbb{R}^2$*, that is*

$$\nu = \sum_{i=1}^{4} \omega_i \delta_{\underline{\boldsymbol{x}}_i}, \qquad \textit{where} \quad \omega_i = \frac{1}{4}, \quad \textit{for} \quad i \in [\![1, 4]\!] \quad \textit{and}$$

$$\underline{\boldsymbol{x}}_1 = [0, 0]^T, \quad \underline{\boldsymbol{x}}_2 = [1, 0]^T, \quad \underline{\boldsymbol{x}}_3 = [0, 1]^T, \quad \underline{\boldsymbol{x}}_4 = [1, 1]^T$$

and let $\Psi = (\psi_1, \psi_2, \psi_3) : \ [0, 1]^2 \to \mathbb{R}^3$*, be given* $\Psi(\underline{\boldsymbol{x}}) = \Psi(x^1, x^2) = [1, x^1, x^2]^T$*. Then the push-forward of* ν *through* Ψ *is given by* $(\Psi_* \nu) = \sum_{i=1}^{4} \omega_i \delta_{\Psi(\underline{\boldsymbol{x}}_i)}$ *and we set*

$$A = [\Psi(\underline{\boldsymbol{x}}_1), \Psi(\underline{\boldsymbol{x}}_2), \Psi(\underline{\boldsymbol{x}}_3), \Psi(\underline{\boldsymbol{x}}_4)] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

and $\mathcal{K}er(A) = span\{\underline{\boldsymbol{\rho}}\} = span\{[1,-1,-1,1]^T\}$. *Since* $rank(A) = 3$ *and* $dim(\mathcal{K}er(A)) = 1$, *we proceed with the* 1*-Tree Measure Reduction. We set* $\underline{\boldsymbol{\lambda}} = \underline{\boldsymbol{\rho}}$ *and compute*

$$\widehat{\alpha} = \min_{1\leq \ell \leq 4}\left\{\frac{\omega_\ell}{\lambda_\ell} \ : \ \lambda_\ell > 0\right\} = \frac{\omega_1}{\lambda_1} = \frac{\omega_4}{\lambda_4} = \frac{1}{4}$$

and similarly if we set $\underline{\boldsymbol{\lambda}} = -\underline{\boldsymbol{\rho}}$ *and compute*

$$\tilde{\alpha} = \min_{1\leq \ell \leq 4}\left\{\frac{\omega_\ell}{\lambda_\ell} \ : \ \lambda_\ell > 0\right\} = \frac{\omega_2}{\lambda_2} = \frac{\omega_3}{\lambda_3} = \frac{1}{4}$$

for each value of $\underline{\boldsymbol{\lambda}}$, *we obtain the corresponding* Ψ*-generalised Carathéodory cubature measure*

$$\nu_{\Psi}^{(1)} = \sum_{i=1}^{2} \widehat{\omega}_i \delta_{\underline{\widehat{\boldsymbol{x}}}_i}, \quad \widehat{\omega}_1 = \widehat{\omega}_2 = \frac{1}{2}, \quad \underline{\widehat{\boldsymbol{x}}}_1 = [1,0]^T, \quad \underline{\widehat{\boldsymbol{x}}}_2 = [0,1]^T$$

$$\nu_{\Psi}^{(2)} = \sum_{i=1}^{2} \tilde{\omega}_i \delta_{\underline{\tilde{\boldsymbol{x}}}_i}, \quad \tilde{\omega}_1 = \tilde{\omega}_2 = \frac{1}{2}, \quad \underline{\tilde{\boldsymbol{x}}}_1 = [0,0]^T, \quad \underline{\tilde{\boldsymbol{x}}}_2 = [1,1]^T$$

In the aforementioned example, the degeneracy for $\nu_{\Psi}^{(1)}$, or equivalently $\nu_{\Psi}^{(2)}$, arises as the result $\mathcal{C}oM(\Psi_*\nu)$ lying on the face of the $Conv\Big(\Psi(\underline{\boldsymbol{x}}_1), \Psi(\underline{\boldsymbol{x}}_2), \Psi(\underline{\boldsymbol{x}}_3)\Big)$ or equivalently on the face of the $Conv\Big(\Psi(\underline{\boldsymbol{x}}_2), \Psi(\underline{\boldsymbol{x}}_3), \Psi(\underline{\boldsymbol{x}}_4)\Big)$.

We can define the face of $Conv\Big(\Psi(\underline{\boldsymbol{x}}_1), \Psi(\underline{\boldsymbol{x}}_2), \Psi(\underline{\boldsymbol{x}}_3)\Big)$, containing $\mathcal{C}oM(\Psi_*\nu)$, as

$$F \ = Conv\Big(\Psi(\underline{\boldsymbol{x}}_1), \Psi(\underline{\boldsymbol{x}}_2), \Psi(\underline{\boldsymbol{x}}_3)\Big) \bigcap \big\{(x,y,z) \in \mathbb{R}^3 \ : \ x = 1, \ 0 \leq y \leq 1, \ z = 1-y\big\}$$

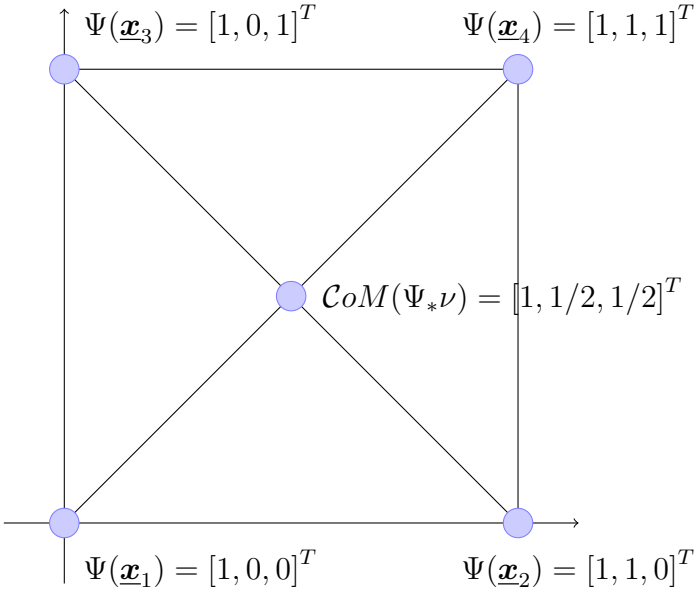


Figure 3.2: Convex Hull Degeneracy

As we can observe in the picture above, we have $F = Conv\Big(\Psi(\underline{\boldsymbol{x}}_2), \Psi(\underline{\boldsymbol{x}}_3)\Big)$ and $\mathcal{C}oM(\Psi_*\nu) \in F$.

Hence, despite the fact that the underlying coefficient matrix A is full-ranked, $rank(A) = M = 3$, the Ψ-generalised "Carathéodory' cubature measure is supported on only $2 = M' < M$ nodes.

To deal with this type of degeneracy, we need to modify the 1-Tree Measure Reduction Algorithm and a similar modification holds for the M-Tree Measure Reduction.

Let $\eta_{M+1} = \sum_{i=1}^{M+1} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$ be given, set $A^{(0)} := [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_{M+1}]$ and assume that $rank(A^{(0)}) = M$. Moreover, let us assume that $\mathcal{C}oM(\eta_{M+1})$ lies on some face of the $Conv(\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_{M+1})$, then $\mathcal{C}oM(\eta_{M+1}) \in Conv(\underline{\boldsymbol{y}}_1, \ldots, \underline{\boldsymbol{y}}_{r^{(1)}-1}, , \underline{\boldsymbol{y}}_{r^{(1)}+1}, \ldots, \underline{\boldsymbol{y}}_{M+1})$ for some $r^{(1)} \in [\![1, M+1]\!]$.

As a result of the above, in step 4) of the 1-Tree Measure Reduction, we will have

$$\alpha_{(1)} := \min_{1 \le j \le (M+1)} \left\{ \frac{\beta_j^{(0)}}{\phi_j} : \ \phi_j > 0 \right\} = \frac{\beta_{k^{(1)}}^{(0)}}{\phi_{k^{(1)}}} = \frac{\beta_{r^{(1)}}^{(0)}}{\phi_{r^{(1)}}}$$
$$\text{for some } k^{(1)}, r^{(1)} \in [\![1, M+1]\!]$$

Hence, we have $(\beta_{k^{(1)}}^{(0)} - \alpha_{(1)}\phi_{k^{(1)}}) = 0$ and $(\beta_{r^{(1)}}^{(0)} - \alpha_{(1)}\phi_{r^{(1)}}) = 0$. Without loss of generality, assume that $k^{(1)} < r^{(1)}$, then in step 5) set $\underline{\boldsymbol{\beta}}^{(1)} := [\beta_1^{(1)}, \beta_2^{(1)}, \ldots, \beta_{(M-1)}^{(1)}]^T$, where

$$\beta_j^{(1)} := \begin{cases} \beta_j^{(0)} - \alpha_{(1)}\phi_j & \text{for } j \in [\![1, k^{(1)} - 1]\!] \\ \beta_{j+1}^{(0)} - \alpha_{(1)}\phi_{j+1} & \text{for } j \in [\![k^{(1)}, r^{(1)} - 2]\!] \\ \beta_{j+2}^{(0)} - \alpha_{(1)}\phi_{j+2} & \text{for } j \in [\![r^{(1)} - 1, M - 1]\!] \end{cases}$$

and similarly in step 6) set $A^{(1)} := [\underline{\boldsymbol{y}}_1^{(1)}, \underline{\boldsymbol{y}}_2^{(1)}, \ldots, \underline{\boldsymbol{y}}_{(M-1)}^{(1)}]$, where

$$\underline{\boldsymbol{y}}_j^{(1)} := \begin{cases} \underline{\boldsymbol{y}}_j^{(0)} & \text{for } j \in [\![1, k^{(i)} - 1]\!] \\ \underline{\boldsymbol{y}}_{j+1}^{(0)} & \text{for } j \in [\![k^{(1)}, r^{(1)} - 2]\!] \\ \underline{\boldsymbol{y}}_{j+2}^{(0)} & \text{for } j \in [\![r^{(1)} - 1, M - 1]\!] \end{cases}$$

This produces, in steps $7) - 8)$ of the 1-Tree Measure Reduction, the reduced weight

vector and coefficient matrix

$$\hat{\underline{\boldsymbol{\beta}}} = [\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_{M-1}]^T := \underline{\boldsymbol{\beta}}^{(1)}$$

$$\hat{A} = [\underline{\boldsymbol{y}}_{i_1}, \underline{\boldsymbol{y}}_{i_2}, \ldots, \underline{\boldsymbol{y}}_{i_{M-1}}] := A^{(1)}$$

and hence we obtain $\tilde{\eta}_{M-1}$ the reduced tree measure for η_{M+1}, given by

$$\tilde{\eta}_{M-1} = \sum_{k=1}^{M-1} \hat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}}$$

3.3.2 Multiplicity of reduced tree measures

As we saw in the two examples in the previous section, the measure reduction algorithms often generate more than one solution. The number of reduced measures $\tilde{\eta}_{M'}$ for η_L, is dependent on the construction of the kernel of the underlying coefficient matrix of η_L. In this section, we make this idea more precise. Let us assume, without loss of generality, that $M^{'} = M$.

Let $\eta_{2M} = \sum_{i=1}^{2M} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$ denote an atomic probability measure on $\mathbb{R}^M$ and set

$$A := [\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_{2M}] \in \mathbb{R}^{M \times 2M}$$
$$\underline{\boldsymbol{\beta}} := [\beta_1, \beta_2, \ldots, \beta_{2M}]^T \in \mathbb{R}_+^{2M}, \ \sum_{i=1}^{2M} \beta_i = 1$$

Then, as noted earlier, producing a reduced measure $\tilde{\eta}_M = \sum_{k=1}^{M} \widehat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}}$ for η_{2M} via the M-Tree Measure Reduction Algorithm, is equivalent, in the language of LP problems, to generating one basic feasible solution $\underline{\widehat{\boldsymbol{\beta}}} = [\widehat{\beta}_1, \ldots, \widehat{\beta}_M, 0, \ldots, 0]^T \in \mathbb{R}_+^{2M}$ to

$$A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}, \text{ where } \quad \underline{\boldsymbol{b}} = \mathcal{C}oM(\eta_{2M}) \in \mathbb{R}^M \tag{3.21}$$
$$\underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}$$

provided that we already have a feasible solution to the aforementioned system, that is we have $\underline{\boldsymbol{\beta}} = [\beta_1, \beta_2, \ldots, \beta_{2M}]^T \in \mathbb{R}_+^{2M}$, $\sum_{i=1}^{2M} \beta_i = 1$ satisfying $A\underline{\boldsymbol{\beta}} = \underline{\boldsymbol{b}}$.

As noted in section (1.3.1), the number of basic feasible solutions or equivalently extreme points of the feasible region $\left\{\underline{\boldsymbol{x}} \in \mathbb{R}^{2M} \ : \ A\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}, \ \ \underline{\boldsymbol{x}} \geq \underline{\mathbf{0}}\right\}$ is upper bounded by $\binom{2M}{M}$. The M-Tree measure reduction can also be utilised to generate more than one basic feasible solution to (3.21). However, by no means, do we guarantee we are able to generate all the basic feasible solutions to (3.21) by applications of the M-Tree measure reduction algorithm. Let S_* denote the set of all basic feasible solutions of (3.21), then we have $|S_*| \leq \binom{2M}{M}$ and let $S \subseteq S_*$ denote the set of basic feasible solutions that can be obtained by several applications of the M-Tree measure reduction. We now describe how to obtain the elements of S.

Since $\mathcal{K}er(A) = span\{\underline{\boldsymbol{\phi}}_1^{(0)}, \underline{\boldsymbol{\phi}}_2^{(0)}, \ldots, \underline{\boldsymbol{\phi}}_M^{(0)}\}$, we clearly have a choice as to the order in which we place the null vectors $\underline{\boldsymbol{\phi}}_i^{(0)}$ into the matrix $\Phi^{(0)}$ and their respective signs. Note that we could also scale any null vector $\underline{\boldsymbol{\phi}}_i^{(0)}$, by a constant $C \in \mathbb{R}$, however this is not desirable since, as illustrated in the following section the null vectors obtained via Singular Value Decomposition form an orthogonal basis for $\mathcal{K}er(A^{(0)})$, and hence

are of norm 1 and for stability purposes of the M-Tree Measure Reduction, we want to keep the norm of the null vectors unaltered.

In the construction of the M-Tree reduction algorithm, we chose $\Phi^{(0)} = [\underline{\boldsymbol{\phi}}_1^{(0)}, \underline{\boldsymbol{\phi}}_2^{(0)}, \ldots, \underline{\boldsymbol{\phi}}_M^{(0)}]$, however many more choices are available. Define the set F_M as follows

$$F_M = \bigcup_{\sigma \in S_M} \bigcup_{n=0}^{2^M-1} \left\{ (-1)^{d_1(n_2)} \underline{\boldsymbol{\phi}}_{\sigma(1)}^{(0)}, (-1)^{d_2(n_2)} \underline{\boldsymbol{\phi}}_{\sigma(2)}^{(0)}, \ldots, (-1)^{d_M(n_2)} \underline{\boldsymbol{\phi}}_{\sigma(M)}^{(0)} \right\}$$

where $d_i : \{0,1\}^M \to \{0,1\}$ selects the i^{th} digit of the binary representation n_2 of the number $n \in [\![0, 2^M - 1]\!]$ and S_M denotes the symmetric group of degree M.

For example, for $M = 2$, we have

$$\begin{aligned} F_2 \quad &= \left\{ \underline{\boldsymbol{\phi}}_1^{(0)}, \underline{\boldsymbol{\phi}}_2^{(0)} \right\} \cup \left\{ \underline{\boldsymbol{\phi}}_2^{(0)}, \underline{\boldsymbol{\phi}}_1^{(0)} \right\} \cup \left\{ \underline{\boldsymbol{\phi}}_1^{(0)}, -\underline{\boldsymbol{\phi}}_2^{(0)} \right\} \cup \left\{ \underline{\boldsymbol{\phi}}_2^{(0)}, -\underline{\boldsymbol{\phi}}_1^{(0)} \right\} \cup \left\{ -\underline{\boldsymbol{\phi}}_1^{(0)}, \underline{\boldsymbol{\phi}}_2^{(0)} \right\} \\ &\cup \left\{ -\underline{\boldsymbol{\phi}}_2^{(0)}, \underline{\boldsymbol{\phi}}_1^{(0)} \right\} \cup \left\{ -\underline{\boldsymbol{\phi}}_1^{(0)}, -\underline{\boldsymbol{\phi}}_2^{(0)} \right\} \cup \left\{ -\underline{\boldsymbol{\phi}}_2^{(0)}, -\underline{\boldsymbol{\phi}}_1^{(0)} \right\} \end{aligned}$$

We can now choose to form $\Phi^{(0)}$ from any element of F_M. We note that not every element of F_M will yield a different basic feasible solution to (3.21) and we cannot determine in advance which choices of $\Phi^{(0)}$ will yield different solutions. We also note that $\left|F_M\right| = 2^M M!$ and hence the number of elements in F_M is greater than the collection of all basic feasible solutions S_*, so clearly a lot of elements in F_M will give the same basic feasible solution.

We note that for any $\Phi^{(0)} \in F_M$, the measure $\tilde{\eta}_M$ produced, corresponding to a certain basic feasible solution, satisfies all the properties in the definition (3.1.9) of a reduced tree measure for η_{2M}. In particular, we note that for any two kernel representations $\Phi_1^{(0)}, \Phi_2^{(0)} \in F_M$, the corresponding reduced tree measures $\tilde{\eta}_{M,1}$ and $\tilde{\eta}_{M,2}$ have the same cardinality, and hence from the perspective of the overall Recombination Algorithm, we cannot make any distinction as to the quality of one reduced tree measure over another.

We also observe that unlike the Simplex Algorithm, we cannot select elements of F_M in order to yield adjacent basic feasible solutions. In fact, if our goal was to construct S_*, then the two-phase Simplex Algorithm would be our preferred method. Recalling the two phase Simplex Algorithm outlined in Section 1.4, the construction of S_* can be achieved as follows: phase (I) - find an initial extreme point solution, phase (II) - move from one extreme point solution to the adjacent one, repeat phase (II) until we have visited every vertex of the feasible region.

By construction of F_M we have no assurance that every vertex of the feasible region will be visited, for all the different kernel constructions $\Phi^{(0)}$, and therefore we cannot insure the ability to construct S*. Consequently, we can only insure that the measure reduction algorithms are suitable for constructing one basic feasible solution to (3.21).

3.3.3 Multiplicity of Ψ-generalised Carathéodory cubature measures

Given an atomic probability measure ν on $\mathbb{R}^d$, with $supp(\nu) \subseteq K$, K compact and a continuous, ν-integrable map $\Psi = (\psi_1, \ldots, \psi_M) : K \to \mathbb{R}^M$, as inputs into the Recombination Algorithm with M-Tree Measure Reduction, we have now identified two sources for producing multiple Ψ-generalised Carathéodory cubature measures ν_Ψ.

The first source is the permutation of leaf nodes in the RHFC Algorithm in step 2) of the Recombination Algorithm. The second source is the existence of multiple reduced tree measures in step 3) of the Recombination Algorithm.

Given the input measure $(\Psi_*\nu)$ with $card\big(supp(\Psi_*\nu)\big) = N$ into the RHFC Algorithm in step 2) of the Recombination Algorithm, denote the output measure from the RHFC Algorithm by $\eta_L^0(\sigma)$, associated with a particular permutation $\sigma \in S_N$ of the leaf nodes. Then, for any $\sigma \in S_N$, $\eta_L^0(\sigma)$ satisfies the definition (3.1.3) of a L-Tree form of $(\Psi_*\nu)$. Moreover, let $\nu_\Psi(\sigma)$ denote the output measure from the Recombination Algorithm, associated with the construction of the RHFC Algorithm of $\eta_L^0(\sigma)$. Then, by definition $\nu_\Psi(\sigma)$ is a Ψ-generalised Carathéodory cubature measure for ν, for any $\sigma \in S_N$.Note, however that it may be possible to find a permutation $\sigma* \in S_N$, yielding

$$card\big(supp(\nu_\Psi(\sigma^*))\big) \leq card\big(supp(\nu_\Psi(\sigma))\big) \qquad \forall \sigma \in S_N$$

Since such σ^* would be unique for the given input measure ν and map $\Psi = (\psi_1, \ldots, \psi_M)$, and the determination of σ^* is prohibitively costly, we do not endeavour to achieve this.

As mentioned, the second source of multiplicity of ν_Ψ, arises in step 3) of the Recombination Algorithm, as a result of multiplicity of reduced tree measures η_M^j for η_{2M}^j. We note that any η_M^j, produced by selecting a different kernel representation $\Phi^{(0)} \in F_M$ in the M-Tree Measure Reduction Algorithm, satisfies the definition (3.1.9) of a reduced tree measure for η_{2M}^j. Moreover, as noted in the previous section, the different kernel representations $\Phi^{(0)} \in F_M$, yield reduced tree measures η_M^j of the same cardinality, hence in terms of the quality of ν_Ψ, the Ψ-generalised Carathéodory cubature measure for ν, produced by the Recombination Algorithm, we make no distinction between different η_M^j .

3.3.4 Stability of the M-Tree Measure Reduction Algorithm

Given the overall computational complexity of the Recombination Algorithm, our measure reduction algorithm of choice is the M-Tree Measure Reduction. The overall Recombination Algorithm is highly sensitive to the numerical computations of the M-Tree Measure Reduction Algorithm, and in particular, the computation of the kernel in step 2). In the following section we examine the numerical stability of this computation.

Given a probability measure $\eta_{2M} = \sum_{i=1}^{2M} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$ on $\mathbb{R}^M$, we proceed to form the matrix $A^{(0)} = [\underline{\boldsymbol{y}}_1, \ldots, \underline{\boldsymbol{y}}_{2M}] \in \mathbb{R}^{M \times 2M}$. To compute the kernel of $A^{(0)}$ numerically, we introduce the singular value decomposition of $A^{(0)}$. For this, we recall the following theorem.

Theorem 3.3.5. *[Theorem 2.4.1 [106]] For any matrix $A \in \mathbb{R}^{M \times N}$, there exist orthonormal matrices*

$$U = [\underline{\boldsymbol{u}}_1, \ldots, \underline{\boldsymbol{u}}_M] \in \mathbb{R}^{M \times M}, \text{ and } V = [\underline{\boldsymbol{v}}_1, \ldots, \underline{\boldsymbol{v}}_N] \in \mathbb{R}^{N \times N},$$

with $U^T U = I$ and $V^T V = I$, such that

$$U^T A V = \Sigma = diag(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{M \times N}, \quad r = min\{M, N\}$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$.

The SVD is a rank-revealing decomposition, as the rank (A) is given as the number of non-zero singular values σ_i in Theorem 3.3.5. Moreover, the SVD provides orthogonal bases for all four fundamental subspaces. Suppose that $rank(A) = r \leq min\{M, N\}$ then the singular value decomposition can be partitioned as follows

$$A = U \Sigma V^T = \begin{bmatrix} U_r & U_0 \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_r^T \\ V_0^T \end{bmatrix} = U_r \Sigma_r V_r^T$$

where $U_r \in \mathbb{R}^{M \times r}$, $V_r \in \mathbb{R}^{N \times r}$ and $\Sigma_r = diag(\sigma_1, \ldots, \sigma_r)$, $\sigma_1 \geq \ldots \geq \sigma_r > 0$, and $\mathcal{C}ol(U_r) = \mathcal{R}an(A)$, $\mathcal{C}ol(U_0) = \mathcal{K}er(A^T)$, $\mathcal{C}ol(V_r) = \mathcal{R}an(A^T)$, $\mathcal{C}ol(V_0) = \mathcal{K}er(A)$.

We know that $rank(A^{(0)}) = M' \leq M$, hence by the aforementioned results, we have

$$A^{(0)} = \begin{bmatrix} U_{M'} & U_0 \end{bmatrix} \left[\begin{array}{cc|c} \Sigma_{M'} & 0 & 0 \ldots 0 \\ 0 & 0 & 0 \ldots 0 \end{array}\right] \begin{bmatrix} V_{M'}^T \\ V_0^T \end{bmatrix} \tag{3.22}$$

where $\Sigma_{M'} = diag(\sigma_1, \ldots, \sigma_{M'})$ with eigenvalues $\sigma_1 \geq \ldots \geq \sigma_{M'} > 0$, whilst $\sigma_{M'+1} = \ldots = \sigma_M = 0$.

Additionally, we note that $dim(\mathcal{K}er(A^{(0)})) = 2M - rank(A^{(0)}) = 2M - M^{'} \geq M$ and we set $\Phi^{(0)} := \mathcal{C}ol(V_0)$ in step 2) of M-Tree Measure Reduction Algorithm, yielding an orthogonal basis of the kernel

$$\Phi^{(0)} = [\underline{\boldsymbol{\phi}}_1^{(0)}, \ldots, \underline{\boldsymbol{\phi}}_{2M-M'}^{(0)}] \in \mathbb{R}^{2M \times (2M-M^{'})}$$

Whilst, the above decomposition $A^{(0)}$ into fundamental subspaces holds true in exact arithmetic, if $A^{(0)}$ is rank-deficient, that is $rank(A^{(0)}) = M^{'} < M$, floating point rounding errors are likely to produce non-zero eigenvalues $\sigma_{M'+1} = \ldots = \sigma_M \neq 0$, which must be recognised as numerical zeros, as stated in the following lemma.

Lemma 3.3.6. *[Lemma 4.2.14 [109]] Suppose $A \in \mathbb{R}^{M \times N}$ has rank $(A) = r$. For every $\epsilon > 0$, there exists a full-rank matrix $A_\epsilon \in \mathbb{R}^{M \times N}$ such that $\|A - A_\epsilon\|_2 = \epsilon$.*

Lemma (3.3.6) states that every rank-deficient matrix has a full-rank matrix arbitrarily close to it. Moreover, the set of full-rank matrices is an open dense subset of $\mathbb{R}^{M \times N}$ and hence its complement is closed and nowhere dense. Hence, given a rank-deficient matrix $A^{(0)}$ in exact arithmetic with $rank(A^{(0)}) = M^{'} < M$, any small perturbation matrix $\Delta A^{(0)}$ occurring due to rounding errors in floating point arithmetic, will almost certainly produce a full-rank matrix $\widehat{A}^{(0)} := A^{(0)} + \Delta A^{(0)}$. Using the matrix approximation $\widehat{A}^{(0)}$ instead of $A^{(0)}$, would lead to overestimating the rank of $A^{(0)}$, and thereeby underestimating the size of the kernel of $A^{(0)}$, as illustrated in the following example.

Let us assume, that in exact precision we are given an underlying coefficient rank-deficient matrix $A^{(0)} = [\underline{\boldsymbol{y}}_1, \ldots, \underline{\boldsymbol{y}}_{2M}] \in \mathbb{R}^{M \times 2M}$ with $rank(A^{(0)}) = M^{'} < M$, then since $A^{(0)}$ has SVD (3.22), we have

$$A^{(0)}\underline{\boldsymbol{v}}_i = \begin{cases} \sigma_i \underline{\boldsymbol{u}}_i & i = 1, \ldots, M^{'} \\ 0 & i = M^{'}+1, \ldots, 2M \end{cases}$$

and consequently the last $(2M - M^{'})$ columns of V from (3.22), form the basis for the kernel, that is

$$\Phi^{(0)} = [\underline{\boldsymbol{\phi}}_1^{(0)}, \ldots, \underline{\boldsymbol{\phi}}_{2M-M'}^{(0)}] = [\underline{\boldsymbol{v}}_{M'+1}, \ldots, \underline{\boldsymbol{v}}_{2M}] \in \mathbb{R}^{2M \times (2M-M^{'})}$$

Now assume that due to floating point precision errors, we observe $\widehat{A}^{(0)} = A^{(0)} + \Delta A^{(0)}$, such that

$$\widehat{A}^{(0)} = \widehat{U} \left[\begin{array}{c|c} \widehat{\Sigma}_M & 0 \ldots 0 \end{array} \right] \widehat{V}^T$$

where $\widehat{\Sigma}_M = diag(\widehat{\sigma}_1, \ldots, \widehat{\sigma}_M)$, with eigenvalues $\widehat{\sigma}_1 \geq \ldots \geq \widehat{\sigma}_M > 0$ and hence conclude that $rank(\widehat{A}^{(0)}) = M$. By the latter assumption, we conclude that the last M columns of $\widehat{V}$ form the basis for the kernel, which we denote by

$$\widehat{\Phi}^{(0)} = [\underline{\widehat{\boldsymbol{\phi}}}_1^{(0)}, \ldots, \underline{\widehat{\boldsymbol{\phi}}}_M^{(0)}] = [\underline{\widehat{\boldsymbol{v}}}_M, \ldots, \underline{\widehat{\boldsymbol{v}}}_{2M}] \in \mathbb{R}^{2M \times M}$$

Since the number of null vectors in the exact arithmetic representation of the kernel $\Phi^{(0)}$ is $(2M - M^{'})$, we would perform the $(2M - M^{'})$-Tree Measure Reduction Algorithm, outlined in Section 3.3.1, this would yield a reduced measure $\tilde{\eta}_{M'}$ for η_{2M}.

In floating point arithmetic representation the number of null vectors $\widehat{\Phi}^{(0)}$ is M and hence we would perform the M-Tree measure Reduction Algorithm, which would yield a reduced measure $\widehat{\eta}_M$ for $\widehat{\eta}_{2M}$.

In the process of the measure reduction in the Recombination Algorithm, we care about two issues: the first is the number of nodes in the reduced measure and the second is centre of mass of the reduced measure. Addressing the first issue, we note that in the above example by utilising the floating point approximation $\widehat{A}^{(0)}$ in place of $A^{(0)}$, we have failed to recognise the last $(M - M^{'})$ eigenvalues as numerical zeros and hence have failed to eliminate an additional $(M - M^{'})$ nodes. In terms of the overall Recombination algorithm this would lead to an increase in the the number of subsequent iterations. Addressing the second issue, requires an estimate of the distance of the centre of mass of the reduced measures $\left\| \mathcal{C}oM(\tilde{\eta}_{M'}) - \mathcal{C}oM(\widehat{\eta}_M) \right\|_2$. As we shall see, we cannot insure much closeness, if the rank of $\widehat{A}^{(0)}$ overestimates the rank of $A^{(0)}$.

Due to the aforementioned issues, we are faced with the problem of numerical rank

determination, which can be stated as follows: suppose that $A^{(0)}$ itself cannot be observed, but rather we are given a perturbed matrix $\widehat{A}^{(0)} := A^{(0)} + \Delta A^{(0)}$, then from the matrix $\widehat{A}^{(0)}$ we wish to extract the rank r of $A^{(0)}$ and find an approximation $\tilde{A}^{(0)}$ to $A^{(0)}$ of rank r. We define the numerical ϵ-rank r_ϵ of a matrix $A^{(0)}$, with respect to some tolerance level $\epsilon > 0$, by

$$r_\epsilon := r_\epsilon(A^{(0)}, \epsilon) = \min_{||\Delta A^{(0)}||_2 \leq \epsilon} rank(A^{(0)} + \Delta A^{(0)}) \tag{3.23}$$

The ϵ-rank of $A^{(0)}$ is equal to the number of columns of $A^{(0)}$ that are guaranteed to be linearly independent for any perturbation of $A^{(0)}$ with norm less than or equal to the tolerance ϵ. In terms of the singular values of $A^{(0)}$, the numerical ϵ-rank r_ϵ satisfies

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{r_\epsilon} > \epsilon \geq \sigma_{r_\epsilon+1}$$

In order to determine the numerical rank of $A^{(0)}$, by observing the perturbed matrix $\widehat{A}^{(0)}$ we need to know how the perturbed eigenvalues $\widehat{\sigma}_i$ of $\widehat{A}^{(0)} = A^{(0)} + \Delta A^{(0)}$ are related to the actual eigenvalues σ_i of $A^{(0)}$. For this, we recall

Lemma 3.3.7. *[Chapter 5.4.1 [106]] If A, $\widehat{A} := A + \Delta A \in \mathbb{R}^{M\times N}$, then*

$$|\sigma_i - \widehat{\sigma}_i| = \left|\sigma_i(A) - \sigma_i(A + \Delta A)\right| \leq \|\Delta A\|_2 \leq \delta||A||_2 \qquad \textit{for } 1 \leq i \leq N$$

The result of the aforementioned lemma is a consequence of backward stability of the *SVD*: the computed *SVD* will be the exact *SVD* of a perturbed matrix $\tilde{A} = \widehat{U}\widehat{\Sigma}\widehat{V}^T = A + \Delta A$, with $||\Delta A||_2 = \delta||A||_2 = \delta\sigma_1$, for some $\delta > 0$, usually taken to represent machine precision. The lemma implies that $\widehat{\sigma}_i$ of $\widehat{A}$ larger than $||\Delta A||_2$ are guaranteed to represent nonzero singular values of A. However, one cannot distinguish the singular values $\widehat{\sigma}_i$ below $||\Delta A||_2$ from a true zero singular value σ_i. As a consequence, when the observed eigenvalues

$$\widehat{\sigma}_k > ||\Delta A||_2 \geq \widehat{\sigma}_{k+1}$$

for some k, one can guarantee that the rank of A is at least k.

Therefore for the problem of numerical rank determination of $A^{(0)}$, assume we have a user-supplied tolerance level $\epsilon \geq ||\Delta A^{(0)}||_2$, usually taken to be consistent with machine precision, e.g. $\epsilon = \delta\|A^{(0)}\|_\infty$, for some machine precision $\delta > 0$. However, if the general level of relative error in the data is larger than δ, e.g if $A^{(0)}$ is correct to 3

digits, ϵ should be correspondingly larger, e.g. $\epsilon = 10^{-3}\|A^{(0)}\|_\infty$. Then, by inspecting the eigenvalues of the observed matrix $\widehat{A}^{(0)}$

$$\widehat{\sigma}_1 \geq \widehat{\sigma}_2 \geq \ldots \geq \widehat{\sigma}_k > \epsilon \geq \widehat{\sigma}_{k+1} \ldots \geq \widehat{\sigma}_M$$

we establish the numerical rank $r_\epsilon = k$ of $A^{(0)}$. We emphasize that the computation of the numerical ϵ-rank is a sensitive computation, thus the amount of confidence we have in the correctness of r_ϵ depends on the underlying gap in the eigenspectrum, between $\widehat{\sigma}_k$ and $\widehat{\sigma}_{k+1}$. And as pointed out in [110] , the notion of numerical ϵ-rank only makes sense in the context of a well-determined gap in the eigenspectrum.
The most common approach to regularisation of a numerically rank-deficient problem is to consider the given matrix $\widehat{A}^{(0)}$, as a noisy representation of the numerically rank-deficient matrix $A^{(0)}$, from which we have extracted the numerical rank $r_\epsilon = k$ and replace $\widehat{A}^{(0)}$ by

$$\widehat{A}_k^{(0)} = \sum_{i=1}^{k} \widehat{\sigma}_i \underline{\widehat{\boldsymbol{u}}}_i \underline{\widehat{\boldsymbol{v}}}_i{}^T$$

the rank-k *TSVD* (Truncated Singular Value Decomposition) of $\widehat{A}^{(0)}$. Essentially, by setting the eigenvalues to zero $\widehat{\sigma}_{k+1} = \ldots = \widehat{\sigma}_M = 0$, we have managed to extract the numerical rank $r_\epsilon = k$ of $A^{(0)}$ from $\widehat{A}^{(0)}$ and construct an approximation to $A^{(0)}$ of rank $r_\epsilon = k$.

We now introduce a measure of distance between the reduced tree measure produced by utilising $A^{(0)}$ in exact arithmetic, to the reduced tree measure produced by utilising $\widehat{A}_{r_\epsilon}^{(0)}$ in floating point arithmetic. In order to derive this result, we recall the following theorem

Theorem 3.3.8 (Eckart-Young Theorem)**.** *[Theorem 2.4.8 [106]] Let $A \in \mathbb{R}^{M\times N}$ with $rank(A) = r$ and let $1 \leq k \leq r$, $k \in \mathbb{N}$ and*

$$A_k = \sum_{i=1}^{k} \sigma_i \underline{\boldsymbol{u}}_i \underline{\boldsymbol{v}}_i{}^T$$

then

$$\min_{rank(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}$$

Theorem 3.3.9. *Let $\eta_{2M} = \sum_{i=1}^{2M} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$ on $\mathbb{R}^M$ be a probability measure and let $\tilde{\eta}_{M'} = \sum_{k=1}^{M'} \widehat{\beta}_k \delta_{\underline{\boldsymbol{y}}_{i_k}}$ denote the reduced measure for η_{2M}. Let the coefficient matrix and probability vector associated with the measure η_{2M}, be given by*

$$A^{(0)} = [\underline{\boldsymbol{y}}_1, \ldots, \underline{\boldsymbol{y}}_{2M}] \in \mathbb{R}^{M \times 2M}$$

$$\underline{\boldsymbol{\beta}}^{(0)} := [\beta_1, \ldots, \beta_{2M}]^T \in \mathbb{R}_+^{2M}$$

and assume that $rank(A^{(0)}) = M' < M$. Let $\widehat{A}^{(0)} = A^{(0)} + \Delta A^{(0)}$, denote a perturbation of $A^{(0)}$ with norm $\|\Delta A^{(0)}\|_2 \leq \epsilon$, for some tolerance level $\epsilon > 0$. Let r_ϵ denote the numerical ϵ-rank of $A^{(0)}$ and let $\widehat{A}_{r_\epsilon}^{(0)}$ denote the rank-r_ϵ TSVD of $\widehat{A}^{(0)}$, given by

$$\widehat{A}_{r_\epsilon}^{(0)} = [\underline{\widehat{\boldsymbol{y}}}_1, \ldots, \underline{\widehat{\boldsymbol{y}}}_{2M}] \in \mathbb{R}^{M \times 2M}$$

then $\widehat{\eta}_{2M} = \sum_{i=1}^{2M} \beta_i \delta_{\underline{\widehat{\boldsymbol{y}}}_i}$ is a probability measure on $\mathbb{R}^M$ and $\widehat{\eta}_{r_\epsilon} = \sum_{k=1}^{r_\epsilon} \widehat{\alpha}_k \delta_{\underline{\widehat{\boldsymbol{y}}}_{i_k}}$ is a reduced measure for $\widehat{\eta}_{2M}$. Then, we have the following bounds

$$\begin{aligned}
&\left\| \mathcal{C}oM(\tilde{\eta}_{M'}) - \mathcal{C}oM(\widehat{\eta}_{r_\epsilon}) \right\|_2 \leq 2\epsilon && \text{if} \quad r_\epsilon < M' \\
&\left\| \mathcal{C}oM(\tilde{\eta}_{M'}) - \mathcal{C}oM(\widehat{\eta}_{r_\epsilon}) \right\|_2 \leq \epsilon && \text{if} \quad r_\epsilon = M' \\
&\left\| \mathcal{C}oM(\tilde{\eta}_{M'}) - \mathcal{C}oM(\widehat{\eta}_{r_\epsilon}) \right\|_2 \leq \left\| A^{(0)} - A_{r_\epsilon}^{(0)} \right\|_2 + \epsilon && \text{if} \quad r_\epsilon > M'
\end{aligned}$$

Proof.

$$\begin{aligned}
&\left\| \mathcal{C}oM(\tilde{\eta}_{M'}) - \mathcal{C}oM(\widehat{\eta}_{r_\epsilon}) \right\|_2 \\
=&\left\| \mathcal{C}oM(\eta_{2M}) - \mathcal{C}oM(\widehat{\eta}_{2M}) \right\|_2 \\
=&\left\| A^{(0)} \underline{\boldsymbol{\beta}}^{(0)} - \widehat{A}_{r_\epsilon}^{(0)} \underline{\boldsymbol{\beta}}^{(0)} \right\|_2 \\
=&\left\| A^{(0)} \underline{\boldsymbol{\beta}}^{(0)} - \left(A_{r_\epsilon}^{(0)} + \Delta A_{r_\epsilon}^{(0)} \right) \underline{\boldsymbol{\beta}}^{(0)} \right\|_2 \\
\leq&\left\| A^{(0)} - A_{r_\epsilon}^{(0)} \right\|_2 \left\| \underline{\boldsymbol{\beta}}^{(0)} \right\|_2 + \left\| \Delta A_{r_\epsilon}^{(0)} \right\|_2 \left\| \underline{\boldsymbol{\beta}}^{(0)} \right\|_2
\end{aligned}$$

If $r_\epsilon \leq M'$, then by Eckart-Young Theorem, we have $\left\| A^{(0)} - A_{r_\epsilon}^{(0)} \right\|_2 = \sigma_{r_\epsilon + 1}$ and since $\left\| \Delta A_{r_\epsilon}^{(0)} \right\|_2 \leq \left\| \Delta A^{(0)} \right\|_2 \leq \epsilon$, $\left\| \underline{\boldsymbol{\beta}}^{(0)} \right\|_2 \leq 1$, we have

$$\left\| \mathcal{C}oM(\tilde{\eta}_{M'}) - \mathcal{C}oM(\widehat{\eta}_{r_\epsilon}) \right\|_2 \leq \sigma_{r_\epsilon + 1} + \epsilon \tag{3.24}$$

By definition of numerical ϵ-rank, we know that $\sigma_{r_\epsilon} > \epsilon \geq \sigma_{r_\epsilon+1}$. Thus, if $r_\epsilon < M^{'}$, the bound is given by

$$\left\| \mathcal{C}oM(\tilde{\eta}_{M^{'}}) - \mathcal{C}oM(\widehat{\eta}_{r_\epsilon}) \right\|_2 \leq \sigma_{r_\epsilon+1} + \epsilon \leq 2\epsilon$$

Whilst, if $r_\epsilon = M^{'}$, then $\sigma_{r_\epsilon+1} = \sigma_{M^{'}+1} = 0$ and we obtain the bound of

$$\left\| \mathcal{C}oM(\tilde{\eta}_{M^{'}}) - \mathcal{C}oM(\widehat{\eta}_{r_\epsilon}) \right\|_2 \leq \sigma_{r_\epsilon+1} + \epsilon \leq \epsilon$$

In the latter case, if $r_\epsilon > M^{'}$, then Eckart-Young Theorem does not apply and we are left with the bound

$$\left\| \mathcal{C}oM(\tilde{\eta}_{M^{'}}) - \mathcal{C}oM(\widehat{\eta}_{r_\epsilon}) \right\|_2 \leq \left\| A^{(0)} - A^{(0)}_{r_\epsilon} \right\|_2 + \epsilon$$

□

Remark 3.3.10.
Theorem 3.3.9 illustrates the primary reason, as to why we are concerned with accurate numerical ϵ-rank determination of the observed constraint matrix $\widehat{A}^{(0)}$.

In the M-Tree Measure Reduction Algorithm, algorithm 6, we implicitly assume that we are operating in infinite precision arithmetic and thus we assume that the equality between the centre of mass of the input measure η_{2M} and the centre of mass of the reduced measure $\tilde{\eta}_{M^{'}}$ is exact, that is

$$A^{(0)}\underline{\boldsymbol{\beta}}^{(0)} = \mathcal{C}oM(\eta_{2M}) = \mathcal{C}oM(\tilde{\eta}_{M^{'}}) = A^{(0)}\underline{\widehat{\boldsymbol{\beta}}}^{(0)}$$

However, since in practice we do not observe $A^{(0)}$ in infinite precision arithmetic, but instead we observe $\widehat{A}^{(0)}$ in finite precision arithmetic, we need to insure that centre of mass of the reduced measure $\tilde{\eta}_{M^{'}}$ in infinite precision arithmetic is sufficiently close to the reduced measure $\widehat{\eta}_{r_\epsilon}$ in finite precision arithmetic. The construction of the latter reduced measure $\widehat{\eta}_{r_\epsilon}$ is dependent on the determination of the numerical ϵ-rank: r_ϵ of the constraint matrix $\widehat{A}^{(0)}$ in finite precision arithmetic.

Theorem 3.3.9 shows that if the numerical ϵ-rank: r_ϵ of $\widehat{A}^{(0)}$ underestimates or estimates exactly the true rank of $A^{(0)}$, then we can ensure that $\mathcal{C}oM(\tilde{\eta}_{M^{'}})$ is sufficiently close to $\mathcal{C}oM(\widehat{\eta}_{r_\epsilon})$, as their distance is bounded above by 2ϵ. This is the best result we can expect to get, as we are determining the numerical ϵ-rank of $\widehat{A}^{(0)}$, based on the

distance between the perturbed and the true singular values, which are also separated by ϵ, as

$$\left|\sigma_i(A^{(0)}) - \sigma_i(\widehat{A}^{(0)})\right| \leq \|\Delta A^{(0)}\|_2 \leq \epsilon$$

However, if the numerical ϵ-rank: r_ϵ of $\widehat{A}^{(0)}$ overestimates the true rank of $A^{(0)}$, then the bound on the distance between $\mathcal{C}oM(\tilde{\eta}_{M'})$ and $\mathcal{C}oM(\widehat{\eta}_{r_\epsilon})$ is $\left\|A^{(0)} - A^{(0)}_{r_\epsilon}\right\|_2 + \epsilon$ and cannot be further improved upon. Therefore, we cannot ensure $\mathcal{C}oM(\tilde{\eta}_{M'})$ and $\mathcal{C}oM(\widehat{\eta}_{r_\epsilon})$ are within ϵ of each other.

The $\|\cdot\|_2$-norm has been chosen to measure the distance between the centres of mass of the reduced measures, as it naturally induces the spectral norm for matrices, and thus enables us to relate the distance of interest to the eigenspectrum of the underlying constraint matrices.

Remark 3.3.11.
As Theorem 3.3.9 and the remark 3.3.10 indicate it is very important not to overestimate the numerical ϵ-rank: r_ϵ of the observed constraint matrix $\widehat{A}^{(0)}$. As illustrated in Lemma 3.3.6, if the exact constraint matrix $A^{(0)}$ is rank-deficient, then the observed constraint $\widehat{A}^{(0)}$ is likely to be full-ranked. Therefore, truncating the smallest singular values at the correct threshold ϵ is crucial. Thus, we need an efficient way to determine the threshold level ϵ. Whilst, there is no universally accepted method in the literature for determining the correct threshold level ϵ, several authors present the following suggestions

1. *In [106], the suggested threshold level is $\epsilon = C\delta\|\widehat{A}^{(0)}\|_1$, where δ represents the machine precision and C represents the general level of relative error in the data and $\|\widehat{A}^{(0)}\|_1 = \max_{1\leq j\leq 2M}\sum_{i=1}^{M}|\widehat{a}_{ij}|$*

2. *In [102], the suggested threshold level is $\epsilon \geq \delta\|\widehat{A}^{(0)}\|_2$, where δ represents the machine precision, $\|\widehat{A}^{(0)}\|_2 = \sigma_{max}(\widehat{A}^{(0)}) = \widehat{\sigma}_1$. The author notes that ϵ may need to be larger, depending on the measure of uncertainty in the data.*

3. *In [108], the author suggests several models for the error $\Delta A^{(0)}$, based on the singular value perturbation estimate*

$$\left|\sigma_i(A^{(0)}) - \sigma_i(\widehat{A}^{(0)})\right| = \left|\sigma_i(A^{(0)}) - \sigma_i(A^{(0)} + \Delta A^{(0)})\right| \leq \|\Delta A^{(0)}\|_2$$

A possible model for the perturbation error the error $\Delta A^{(0)}$, is a random matrix with elements from a certain statistical distribution.

(a) $\Delta A_N^{(0)} \sim \mathcal{N}(0, \sigma^2)$

(b) $\Delta A_P^{(0)} \sim Poisson(\lambda)$

with estimates $\mathbb{E}[\Delta A_N^{(0)}]$ *for* $\Delta A_N^{(0)}$ *and* $\mathbb{E}[\Delta A_P^{(0)}]$ *for* $\Delta A_P^{(0)}$ *given by*

(a) $\mathbb{E}[\|\Delta A_N^{(0)}\|] = \sigma\sqrt{2M}$

(b) $\mathbb{E}[\|\Delta A_P^{(0)}\|] = \lambda\sqrt{2M^2}$

Similar estimates are also derived in [108] for the Frobenius norm.

As noted in [112], in order for definitions of numerical ϵ-rank to make sense, we must assume that the underlying coefficient matrix $A^{(0)}$ has been properly scaled. A good scaling strategy ensures that either the columns of the matrix $A^{(0)}$ have approximately the same norm or that the uncertainties in all elements of $A^{(0)}$ are of the same order of magnitude. We note however that both row and column scalings of $A^{(0)}$ have a considerable effect on its eigenspectrum. For a full-rank matrix $A^{(0)}$ one can equilibrate the rows or columns of $A^{(0)}$, by diagonally scaling $A^{(0)}$ from left or right, such that all the rows and columns of the scaled matrix have equal norm; optimal diagonal scalings are related to the elements of the first and the last elements of U and V in the *SVD* of $A^{(0)}$ as noted in Chapter 3 of [108]. Whilst, for rank-deficient problems, scaling may present problems, as noted in [100]. Hence, it is essential for the stability of the algorithm and detection of numerical rank to ensure that the underlying matrix is properly scaled.

Let us now examine, the column scaling of the matrix $A^{(0)}$ in the context of the overall Recombination Algorithm with M-Tree Measure Reduction. We recall that given the input measure into the Recombination Algorithm with M-Tree Measure Reduction

$$\nu = \sum_{i=1}^{N} \omega_i \delta_{\underline{\boldsymbol{x}}_i}, \qquad \underline{\boldsymbol{x}}_i \in K, \text{ for some } K \in \mathcal{B}(\mathbb{R}^d) \text{ compact}$$

and in step 2) of the Recombination Algorithm with M-Tree Measure Reduction we produce the push-forward of ν through $\Psi := (\psi_1, \ldots, \psi_M) : \ K \to \mathbb{R}^M$, given by

$$(\Psi_* \nu) = \sum_{i=1}^{N} \omega_i \delta_{\Psi(\underline{\boldsymbol{x}}_i)}, \qquad \Psi(\underline{\boldsymbol{x}}_i) \in \Psi(K) \subset \mathbb{R}^M$$

subsequently in step 3) we form $2M$-Tree form of $(\Psi_* \nu)$, via the RHFC Algorithm, given by

$$\eta_{2M} := \eta_{2M}^0 = \sum_{i=1}^{2M} \beta_i \delta_{\underline{\boldsymbol{y}}_i}$$

we recall from definition (3.1.3), that

$$supp(\eta_{2M}) \subseteq Conv\big(\{\Psi(\underline{\boldsymbol{x}}_1), \ldots, \Psi(\underline{\boldsymbol{x}}_N)\}\big)$$

We recall from remark (3.1.7),

$$\underline{\boldsymbol{y}}_i = \begin{cases} \underline{\boldsymbol{x}}_i^D & \text{for} \quad 1 \leq i \leq r \\ \underline{\boldsymbol{x}}_{p+i-(r+1)}^{D-1} & \text{for} \quad r+1 \leq i \leq 2M \end{cases}$$

where $r = \Big(\lceil \frac{N}{2^{D-1}} \rceil - L\Big)$ and $p = 2r$ if $\lceil \frac{N}{2^{D-1}} \rceil = 2\ell + 1$, $p = 2r + 1$ if $\lceil \frac{N}{2^{D-1}} \rceil = 2\ell$, $\ell \in \mathbb{N}$. Assume, without loss of generality, for any $k, \ell \in [\![1, 2M]\!]$

$$\underline{\boldsymbol{y}}_k = \underline{\boldsymbol{x}}_k^D \quad \text{and} \quad \underline{\boldsymbol{y}}_\ell = \underline{\boldsymbol{x}}_\ell^D$$

Recalling the expressions for $\underline{\boldsymbol{x}}_k^D$, $\underline{\boldsymbol{x}}_\ell^D$, we have

$$\|\underline{\boldsymbol{y}}_k\|_1 = \sum_{j=1}^{M} |y_{k,j}| \leq \sum_{j=1}^{M} \Big(\sum_{r=(k-1)2^D+1}^{k2^D} \omega_{k,r}^{'} |\psi_j(\underline{\boldsymbol{x}}_r)| \Big) \leq \sum_{j=1}^{M} C_j$$

$$\|\underline{\boldsymbol{y}}_\ell\|_1 = \sum_{j=1}^{M} |y_{\ell,j}| \leq \sum_{j=1}^{M} \Big(\sum_{r=(\ell-1)2^D+1}^{\ell 2^D} \omega_{\ell,r}^{'} |\psi_j(\underline{\boldsymbol{x}}_r)| \Big) \leq \sum_{j=1}^{M} C_j$$

since

$$\sum_{r=(k-1)2^D+1}^{k2^D} \omega_{k,r}^{'} = 1, \qquad \sum_{r=(\ell-1)2^D+1}^{\ell 2^D} \omega_{\ell,r}^{'} = 1,$$

and by assumption on ψ_j, there exists $C_j \in \mathbb{R}_+$, $\forall j \in [\![1, 2M]\!]$, such that

$$\sup_{\underline{\boldsymbol{x}} \in K} \big|\psi_j(\underline{\boldsymbol{x}})\big| = C_j$$

since a continuous image of a compact set $K \subset \mathbb{R}^d$ is compact. Depending on the size of C_j, the aforementioned bounds may not be sufficient to conclude that

$$\|\underline{\boldsymbol{y}}_k\|_1 \approx \|\underline{\boldsymbol{y}}_\ell\|_1$$

and hence conclude that the matrix $A^{(0)}$ is scaled appropriately. We note that a similar type of analysis can be carried out for subsequent measures η_{2M}^j, for $j \geq 1$, produced by the Recombination Algorithm.

By scaling the input map Ψ into the Recombination Algorithm, so as to obtain $\tilde{\Psi} = (\tilde{\psi}_1, \ldots, \tilde{\psi}_M) : K \to \mathbb{R}^M$, where

$$\tilde{\psi}_j := \frac{\psi_j}{C_j}$$

we would ensure that each element of the matrix $A^{(0)}$, satisfies

$$|y_{k,j}| \leq \sum_{r=(k-1)2^D+1}^{k2^D} \omega'_{k,r} |\psi_j(\underline{\boldsymbol{x}}_r)| \leq 1$$

This ensures the stability of the M-Tree Measure Reduction Algorithm, as every entry of the input coefficient matrix $A^{(0)} = [\underline{\boldsymbol{y}}_1, \ldots, \underline{\boldsymbol{y}}_{2M}]$ is of the same order of magnitude. However, this approach would only produce a $\tilde{\Psi}$-generalised cubature measure for ν, that is a measure $\nu_{\tilde{\Psi}}$, satisfying

$$1) \ \int_K \tilde{\psi}_j(\underline{\boldsymbol{x}}) d\nu(\underline{\boldsymbol{x}}) = \int_K \tilde{\psi}_j(\underline{\boldsymbol{x}}) d\nu_{\tilde{\Psi}}(\underline{\boldsymbol{x}}) \quad \forall j \in [\![1, M]\!]$$

$$2) \ supp(\nu_{\tilde{\Psi}}) \subseteq supp(\nu)$$

and hence we would only have

$$\int_K \psi_j(\underline{\boldsymbol{x}}) d\nu(\underline{\boldsymbol{x}}) = C_j \int_K \tilde{\psi}_j(\underline{\boldsymbol{x}}) d\nu_{\tilde{\Psi}}(\underline{\boldsymbol{x}}) \quad \forall j \in [\![1, M]\!]$$

The column scaling of the underlying matrix $A^{(0)}$, in this general setting of an arbitrary Borel compact set K and a continuous map Ψ, requires further research and investigation, outside the scope of this dissertation.

In the following chapter, we shall specialise the Recombination Algorithm to computing Carathéodory cubature measures for a certain class of positive Borel measures μ on $[0,1]^d$. In that context, we will illustrate the practical implementation of the Recombination algorithm, by taking $K = [0,1]^d$ and specialising the map $\Psi := \gamma_m^d$, where

$$\gamma_m^d := (\gamma_1, \ldots, \gamma_M) : [0,1]^d \to \mathbb{R}^M$$

to be the (m,d)-monomial map, introduced in definition (2.0.22), for some fixed basis $\mathcal{B}(\Pi_m^d) = \{\gamma_1, \ldots, \gamma_M\}$. In our implementation, we shall employ the orthogonal basis of Chebyshev polynomials of first kind, scaled onto the unit cube. Under this setting, each element of the underlying coefficient matrix $A^{(0)}$ satisfies

$$|y_{k,j}| \leq \sum_{r=(k-1)2^D+1}^{k2^D} \omega'_{k,r} |\gamma_j(\underline{\boldsymbol{x}}_r)| \leq 1$$

and empirically we find that the coefficient matrix $A^{(0)}$, is well-scaled, in the sense that $A^{(0)}$ exhibits a prominent gap between r_ϵ and $r_{\epsilon+1}$ and hence we have a good level of confidence in determining the numerical rank of $A^{(0)}$, through the eigenspectrum of $\widehat{A}^{(0)}$.

3.3.4.1 Numerical ϵ-rank and Eigenspectrum of rank-deficient matrices

In the following we present two examples of the distribution of eigenvalues around the numerical ϵ-rank of a rank-deficient constraint matrix $\widehat{A}^{(0)} \in \mathbb{R}^{M\times 2M}$. These examples were drawn from a large set of Gauss-Legendre Carathéodory cubature computations, that will be illustrated in Chapter 5, see Table 5.9. The two examples we are about to illustrate, were chosen as examples of clearly rank-deficient constraint matrices, as they exhibit a prominent gap in the eigenspectrum. Other examples in the same problem set are classed as full-ranked and others still exhibit a less prominent gap in the eigenspectrum, making it less evident as to whether the matrix should be considered rank-deficient. All the computations are carried out in double precision.

Heuristically, after some numerical experimentation, we employ a variant of the suggested ϵ-threshold in [106], referenced in Remark 3.3.11, to determine the numerical ϵ-rank. Namely, we employ

$$\epsilon = C\delta\|\widehat{A}^{(0)}\|_\infty$$

where $C = 10^2$ to account for the level of uncertainty in the data, this may need to be increased dependent on problem size, $\delta = 10^{-16}$ is the double machine precision and $\|\widehat{A}^{(0)}\|_\infty = \max_{1\leq i\leq M}\sum_{j=1}^{2M}|\widehat{a}_{ij}|$ is the maximum absolute row sum of the matrix. The matrix norm: $\|\widehat{A}^{(0)}\|_\infty$ grows linearly with M and heuristically appears to be a good measure for relating the ϵ-threshold to the size of the matrix.

The input cubature data into the Recombination Algorithm is scaled onto $[0,1]^d$. This scaling is crucial, as it ensures that the monomial map $\gamma_m^d := (\gamma_1,\ldots,\gamma_M) : [0,1]^d \to \mathbb{R}^M$, mapping the input cubature data onto matrix entries $\widehat{a}_{ij}$ of $\widehat{A}^{(0)}$, yields $|\widehat{a}_{ij}| \leq 1$. Therefore, the resulting constraint matrix $\widehat{A}^{(0)}$ has rows of similar magnitude and can be considered well-scaled for the purposes of numerical ϵ-rank determination.

In the first example we take

$$M = 1820,\ r_\epsilon = 1663$$

$$\epsilon := 10^2 \delta \|\widehat{A}^{(0)}\|_\infty$$

$$\delta = 10^{-16},\ \|\widehat{A}^{(0)}\|_\infty = 940.4$$

and in the second

$$M = 2390,\ r_\epsilon = 2198$$

$$\epsilon := 10^2 \delta \|\widehat{A}^{(0)}\|_\infty$$

$$\delta = 10^{-16},\ \|\widehat{A}^{(0)}\|_\infty = 1132.3$$

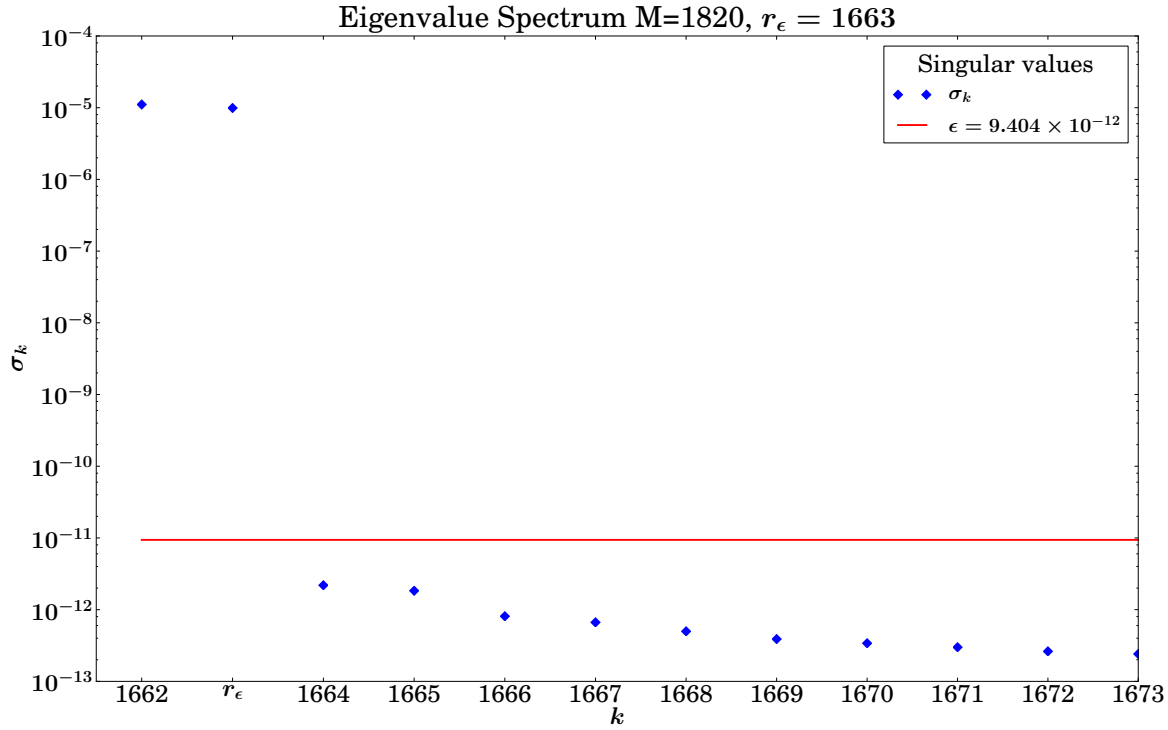


Figure 3.3: Eigenspectrum Figure 1

k	1659	1660	1661	1662	1663
σ_k	0.000603366	0.000341535	0.000239577	0.000011079	0.0000099

k	1664	1665	1666	1667	1668
σ_k	2.18964×10^{-12}	1.83492×10^{-12}	8.90199×10^{-13}	$6.6529\ \times10^{-13}$	4.98587×10^{-13}

k	1669	1670	1671	1672	1673
σ_k	3.88926×10^{-13}	3.40067×10^{-13}	2.99397×10^{-13}	2.62083×10^{-13}	2.41040×10^{-13}

Table 3.1: Eigenspectrum table 1

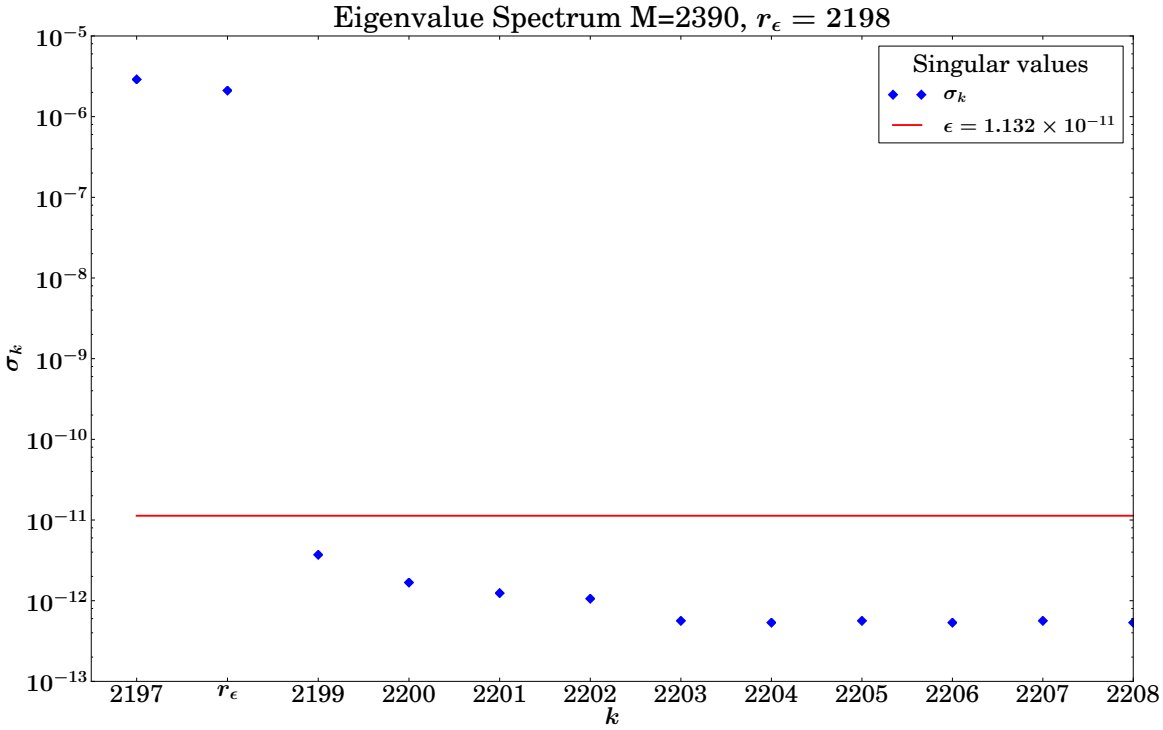


Figure 3.4: Eigenspectrum Figure 2

k	2194	2195	2196	2197	2198
σ_k	3.78996×10^{-5}	2.89475×10^{-5}	1.38641×10^{-5}	2.89288×10^{-6}	2.1086×10^{-6}

k	2199	2200	2201	2202	2203
σ_k	3.70915×10^{-12}	1.6799×10^{-12}	1.24132×10^{-12}	1.05967×10^{-12}	5.6331×10^{-13}

k	2204	2205	2206	2207	2208
σ_k	5.34936×10^{-13}	4.5082×10^{-13}	4.08371×10^{-13}	3.51395×10^{-13}	3.31479×10^{-13}

Table 3.2: Eigenspectrum table 2

Chapter 4

Carathéodory cubature measures

As we saw in the previous chapter, given an atomic probability measure $\nu \in \mathcal{M}_+(\mathbb{R}^d)$ with $supp(\nu) \subseteq K$ and $\Psi = (\psi_1, \ldots, \psi_M) : \ K \to \mathbb{R}^M$ a continuous, ν-integrable map, we can exploit the Recombination Algorithm to construct a Ψ-generalised Carathéodory cubature measure $\widehat{\nu}_\Psi \in \mathcal{M}_+(\mathbb{R}^d)$. In the forthcoming chapter, building on the ideas introduced in Chapter 3, we shall specialise the Recombination Algorithm to the computation of Carathéodory cubature measure $\hat{\nu}_m^\mu$ on $[0,1]^d$ for a certain class of positive Borel measures μ on $[0,1]^d$ and extend the underlying algorithm by introducing further recursive methods, to accelerate the overall computation.

In the following, we shall restrict our attention to a class of positive Borel measures μ on $[0,1]^d$, admitting a factorable density ρ, that is $\mu(A) = \int_A \rho(\underline{\boldsymbol{x}}) d\underline{\boldsymbol{x}}$, $A \subseteq [0,1]^d$, such that density ρ can be factored into a product of univariate densities ρ_i, such that

$$\rho(x^1, \ldots, x^d) = \prod_{i=1}^{d} \rho_i(x^i), \ \ \rho_i : [0,1] \to \mathbb{R}_+$$

and we shall further require that quadrature measures of arbitrary degree $m \in \mathbb{N}$ for densities ρ_i on $[0,1]$, for $1 \leq i \leq d$, are known.

We note that whilst, the aforementioned class of measures is small, it contains several important examples. The simplest measure satisfying the above conditions is given by the d-dimensional Lebesgue measure, for which the univariate densities are given by $p_i(x^i) = 1$ for $1 \leq i \leq d$ and Gauss-Legendre quadratures are known. Another example is given by taking density $\rho_i(x^i) = (1-x^i)^\alpha (x^i)^\beta$ for $\alpha, \beta > -1$ for $1 \leq i \leq d$, for which the Gauss-Jacobi quadratures, scaled onto the unit interval, are also known, see in [111]. In fact, we can take ρ_i to be the weight function, scaled onto the unit

interval, of any interpolatory quadrature on a compact interval, such as those mentioned in Chapter 1.

By assumption univariate quadrature formulae of arbitrary degree $m \in \mathbb{N}$ on $[0,1]$ are available for densities ρ_i, $i \in [\![1,d]\!]$. Hence, we have a collection of quadrature formulae $\{\tilde{Q}_1, \ldots, \tilde{Q}_d\}$ of some degree m, given by

$$\tilde{Q}_i = \sum_{j=1}^{n} \tilde{\lambda}_k^i \delta_{x_k^i} \qquad \tilde{\lambda}_k^i \in \mathbb{R}_+, \quad \{x_k^i\}_{k=1}^n \subset [0,1] \quad \text{for} \quad i \in [\![1,d]\!] \tag{4.1}$$

satisfying

$$\int_{[0,1]} \pi_j(x^i)\rho^i(x^i)dx^i = \sum_{k=1}^{n} \tilde{\lambda}_k^i \pi_j(x_k^i), \quad \text{for } j \in [\![1,m]\!],\ i \in [\![1,d]\!]$$

for some fixed basis for Π_m^1, given by $\mathcal{B}(\Pi_m^1) = \{\pi_1, \ldots, \pi_{m+1}\}$.

As noted in Chapter 1, we can always construct the Cartesian product cubature measure ζ_m^μ of degree m for μ from $\{\tilde{Q}_1, \ldots, \tilde{Q}_d\}$, that is

$$\begin{aligned}
\zeta_m^\mu &= \tilde{Q}_1 \otimes \ldots \otimes \tilde{Q}_d \\
&= \sum_{k_1=1}^{n} \tilde{\lambda}_{k_1}^1 \delta_{x_{k_1}^1} \otimes \ldots \otimes \sum_{k_d=1}^{n} \tilde{\lambda}_{k_d}^1 \delta_{x_{k_d}^1} \\
&= \sum_{k_1=1}^{n} \ldots \sum_{k_d=1}^{n} \tilde{\lambda}_{k_1}^1 \ldots \tilde{\lambda}_{j_d}^d \delta_{(x_{k_1}^1, \ldots, x_{k_d}^d)}
\end{aligned}$$

satisfying

$$\begin{aligned}
\mathbb{E}_\mu[\gamma_j] &= \int_{[0,1]^d} \gamma_j(x^1, \ldots, x^d) d\mu(x^1, \ldots, x^d) \\
&= \int_{[0,1]^d} \gamma_j(x^1, \ldots, x^d) \prod_{i=1}^{d} \rho_i(x^i) dx^1 \ldots dx^d \\
&= \sum_{k_1=1}^{n} \ldots \sum_{k_d=1}^{n} \tilde{\lambda}_{k_1}^1 \ldots \tilde{\lambda}_{k_d}^d \gamma_j(x_{k_1}^1, \ldots, x_{k_d}^d) = \mathbb{E}_{\zeta_m^\mu}[\gamma_j]
\end{aligned} \tag{4.2}$$

for some fixed basis for Π_m^d, given by $\mathcal{B}(\Pi_m^d) = \{\gamma_1, \ldots, \gamma_M\}$, $M := dim(\Pi_m^d) = \binom{m+d}{d}$.

However, for computational purposes, instead of utilising the univariate quadratures $\tilde{Q}_i$ directly to approximate the integral with respect to μ in (4.2), we proceed to normalise each quadrature $\tilde{Q}_i$ in (4.1) to one. Thus, setting

$$\lambda_k^i := \frac{\tilde{\lambda}_k^i}{\sum_{i=1}^n \tilde{\lambda}_k^i}$$

we obtain

$$Q_i = \sum_{k=1}^{n} \lambda_k^i \delta_{x_k^i} \qquad \lambda_k^i \in \mathbb{R}_+, \ \sum_{i=1}^{n} \lambda_k^i = 1, \quad \left\{x_k^i\right\}_{k=1}^n \subset [0,1] \quad \text{for} \ \ i \in [\![1,d]\!]$$

Setting $\nu_m^\mu = Q_1 \otimes \ldots \otimes Q_d$, for $\gamma_j \in \mathcal{B}(\Pi_m^d)$, we get

$$\begin{aligned}\mathbb{E}_\mu[\gamma_j] &= \mu([0,1]^d) \sum_{k_1=1}^{n} \ldots \sum_{k_d=1}^{n} \lambda_{k_1}^1 \ldots \lambda_{k_d}^d \gamma_j(x_{k_1}^1, \ldots, x_{k_d}^d) \\ &= \mu([0,1]^d) \mathbb{E}_{\nu_m^\mu}[\gamma_j] = \mathbb{E}_{\zeta_m^\mu}[\gamma_j]\end{aligned}$$

Hence, we have $\zeta_m^\mu = \mu([0,1]^d)\nu_m^\mu$ is a cubature measure of degree m for μ, since by construction

1) $\displaystyle\int_{[0,1]^d} \gamma_j(\underline{\boldsymbol{x}}) d\mu(\underline{\boldsymbol{x}}) = \int_{[0,1]^d} \gamma_j(\underline{\boldsymbol{x}}) d\zeta_m^\mu(\underline{\boldsymbol{x}}) = \mu([0,1]^d) \int_{[0,1]^d} \gamma_j(\underline{\boldsymbol{x}}) d\nu_m^\mu(\underline{\boldsymbol{x}}) \quad \forall j \in [\![1,M]\!]$

2) $supp(\zeta_m^\mu) \subseteq supp(\mu)$

Now, suppose that we wish to construct a Carathéodory cubature measure of degree m for μ, that is a measure $\widehat{\zeta}_m^\mu \in \mathcal{M}_+([0,1]^d)$, satisfying

1) $\displaystyle\int_K \gamma_j(\underline{\boldsymbol{x}}) d\mu(\underline{\boldsymbol{x}}) = \int_K \gamma_j(\underline{\boldsymbol{x}}) d\widehat{\zeta}_m^\mu(\underline{\boldsymbol{x}}) \quad \forall j \in [\![1,M]\!]$

2) $supp(\widehat{\zeta}_m^\mu) \subseteq supp(\mu)$

3) $card\big(supp(\widehat{\zeta}_m^\mu)\big) \leq M+1$

Applying the Recombination Algorithm with M-Tree Measure Reduction, we set the input measure $\nu := \nu_m^\mu$ and $\Psi := \gamma_m^d$, where $\gamma_m^d := (\gamma_1, \ldots, \gamma_M) : [0,1]^d \to \mathbb{R}^M$ denotes the (m,d)-monomial mapping, introduced in definition (2.0.22) in Chapter 2. Then, as output we produce $\widehat{\nu}_m^\mu$, a Carathéodory cubature measure of degree m for ν_m^μ. And therefore, by setting $\widehat{\zeta}_m^\mu := \mu([0,1]^d)\widehat{\nu}_m^\mu$, we obtain a Carathéodory cubature measure of degree m for μ.

However, we note that the construction of $\widehat{\nu}_m^\mu$ leads to touching a number of nodes, which grows exponentially with dimension d, in the construction of ν_m^μ. Hence, in

order to avoid this curse of dimensionality, we will employ a recursive algorithm. In this algorithm we will construct the tensor product measure one dimension at a time and reduce it via the Recombination Algorithm, in order to keep the size of each underlying problem tractable.

In the following, we let $\nu_{m,\ell}^{\mu}$, $\widehat{\nu}_{m,\ell}^{\mu}$ denote, respectively, a cubature measure of degree m for μ_ℓ in ℓ dimensions and a Carathéodory cubature measure of degree m for μ_ℓ in ℓ dimensions, where μ_ℓ is given by

$$\mu_\ell(A) = \int_A \rho_1(x^1) \times \ldots \times \rho_\ell(x^\ell) dx^1 \ldots dx^\ell, \qquad A \subset [0,1]^\ell, \qquad \ell \leq d$$

The recursive algorithm for constructing $\widehat{\nu}_m^{\mu}$ may be defined as follows

Set $\widehat{\nu}_{m,1}^{\mu} = \nu_{m,1}^{\mu}$ and

For $\ell = 1 : d-1$

1) Compute the cubature measure $\nu_{m,\ell+1}^{\mu}$ of degree m for $\mu_{\ell+1}$, given by

$$\begin{aligned} &\nu_{m,\ell+1}^{\mu} = \widehat{\nu}_{m,\ell}^{\mu} \otimes \nu_{m,1}^{\mu}, \quad card\Big(supp(\nu_{m,\ell+1}^{\mu})\Big) = M_\ell^{'} \times n \\ &\text{where} \quad M_\ell^{'} = card\Big(supp(\nu_{m,\ell}^{\mu})\Big) \leq \left|\Gamma_m^\ell[\mathbb{R}]\right| = \binom{m+\ell}{\ell} \end{aligned}$$

2) Compute the Carathéodory cubature measure $\widehat{\nu}_{m,\ell+1}^{\mu}$ of degree m for $\mu_{\ell+1}$, by an application of the Recombination Algorithm with M_ℓ-Tree Measure Reduction to $\nu_{m,\ell+1}^{\mu}$ and set $\Psi := (\gamma_1, \ldots, \gamma_{M_{\ell+1}})$, for some fixed basis $\mathcal{B}(\Pi_m^{\ell+1}) = \left\{\gamma_1, \ldots, \gamma_{M_{\ell+1}}\right\}$

$$\begin{aligned} &\nu_{m,\ell+1}^{\mu} \to \widehat{\nu}_{m,\ell+1}^{\mu}, \quad card\Big(supp(\widehat{\nu}_{m,\ell+1}^{\mu})\Big) = M_{\ell+1}^{'} \\ &\text{where} \quad M_{\ell+1}^{'} = card\Big(supp(\nu_{m,\ell+1}^{\mu})\Big) \leq \left|\Gamma_m^{\ell+1}[\mathbb{R}]\right| = \binom{m+\ell+1}{\ell+1} \end{aligned}$$

This recursive algorithm by dimension, would generate the Carathéodory cubature measure $\widehat{\nu}_m^{\mu} = \widehat{\nu}_{m,d}^{\mu}$ of degree m for ν_m^{μ} in polynomial rather than exponential time

with dimension. However, there are some subtleties, and some modifications are required.

In step 2) of the algorithm, applying the Recombination Algorithm with M_ℓ-Tree Measure Reduction with input measure $\nu^{\mu}_{m,\ell+1}$, requires the cardinality of the input measure to satisfy

$$card\big(supp(\nu^{\mu}_{m,\ell+1})\big) = M'_\ell \times n > 2M_{\ell+1} \tag{4.3}$$

However, (4.3) may not hold for sufficiently small degree m and dimension $\ell+1$, as will be discussed in detail in this section. Under this scenario, we cannot apply the Recombination Algorithm with M_ℓ-Tree Measure Reduction, as step 2) of the Recombination Algorithm would require to build the $2M_{\ell+1}$-Tree form of $\nu^{\mu}_{m,\ell+1}$ via the RHFC Algorithm and this cannot be done with less than $2M_{\ell+1}$ nodes in the support of the input measure. Hence a modification of the Recombination Algorithm with M_ℓ-Tree Measure Reduction will be required to deal with the case, when (4.3) does not hold. This modification shall be discussed in the forthcoming algorithm.

Secondly, we set $\Psi := \gamma^{\ell+1}_m$, where $\gamma^{\ell+1}_m$ is $(m, \ell+1)$-monomial map:

$$\gamma^{\ell+1}_m := (\gamma_1, \ldots, \gamma_{M_{\ell+1}}) : [0,1]^{\ell+1} \to \mathbb{R}^{M_{\ell+1}}$$

Heuristically, we decided to employ the orthogonal basis $\{\gamma_1, \ldots, \gamma_{M_{\ell+1}}\}$ of Chebyshev polynomials of first kind, scaled onto the unit cube, for its well-known stability properties. The first few Chebyshev polynomials in 1 dimension, scaled onto the unit interval, are given by

$$T_1(x) = 1$$

$$T_2(x) = 2x - 1$$

$$T_3(x) = 8x^2 - 8x + 1$$

$$T_4(x) = 32x^3 - 48x^2 + 18x - 1$$

$$T_5(x) = 128x^4 - 256x^3 + 160x^2 - 32x + 1$$

Hence, for example the $(2,2)$-monomial map: γ^2_2 is given by

$$\gamma^2_2(\underline{\boldsymbol{x}}) = \gamma^2_2(x^1, x^2) = [1, 2x^1 - 1, 2x^2 - 1, 8(x^1)^2 - 8x^1 + 1, 4x^1x^2 - 2x^1 - 2x^2 + 1, 8(x^2)^2 - 8x^2 + 1]^T$$

Then, we can define the $(m, \ell+1)$-monomial form of $\nu^{\mu}_{m,\ell+1}$, introduced in (2.0.23), as follows

$$\gamma^{\mu}_{m,\ell+1} := {\gamma^{\ell+1}_{m}}_{*}(\nu^{\mu}_{m,\ell+1})$$

Now, we are in a position to describe the construction of the Carathéodory Cubature Algorithm.

Algorithm 8: Carathéodory Cubature Algorithm

Input: $\nu_{m,1}^{\mu}$ cubature measure of degree $m \in \mathbb{N}$ for μ_1

1) Set $\widehat{\nu}_{m,1}^{\mu} := \nu_{m,1}^{\mu}$

For $\ell = 1 : d-1$

2) Compute the cubature measure of degree m, $\nu_{m,\ell+1}^{\mu}$ for $\mu_{\ell+1}$:

$$\nu_{m,\ell+1}^{\mu} = \widehat{\nu}_{m,\ell}^{\mu} \otimes \nu_{m,1}^{\mu} = \sum_{i=1}^{N_{\ell+1}} \omega_i \delta_{\underline{\boldsymbol{x}}_i}$$

and set

$$N_{\ell+1} := supp(\nu_{m,\ell+1}^{\mu}) = M_{\ell}^{'} \times card\big(supp(\nu_{m,1}^{\mu})\big), \quad M_{\ell+1} := \tbinom{m+\ell+1}{\ell+1}$$

If $\Big(\min\big\{N_{\ell+1}, 2M_{\ell+1}\big\} = N_{\ell+1}\Big)$

3) Produce the $(m, \ell+1)$-monomial form of $\nu_{m,\ell+1}^{\mu}$:

$$\gamma_{m,\ell+1}^{\mu} := {\gamma_m^{\ell+1}}_{*}(\nu_{m,\ell+1}^{\mu}) = \sum_{i=1}^{N_{\ell+1}} \omega_i \delta_{\gamma_m^{\ell+1}(\underline{\boldsymbol{x}}_i)}$$

4) Produce a reduced tree measure $\widehat{\gamma}_{m,\ell+1}^{\mu}$ for $\gamma_{m,\ell+1}^{\mu}$:

$$\widehat{\gamma}_{m,\ell+1}^{\mu} = \sum_{k=1}^{M_{\ell+1}^{'}} \widehat{\omega}_{i_k} \delta_{\gamma_m^{\ell+1}(\underline{\boldsymbol{x}}_{i_k})}$$

where $M_{\ell+1}^{'} \leq M_{\ell+1}$, and produce the Carathéodory cubature measure of degree m, $\widehat{\nu}_{m,\ell+1}^{\mu}$ for $\mu_{\ell+1}$

$$\widehat{\nu}_{m,\ell+1}^{\mu} = \sum_{k=1}^{M_{\ell+1}^{'}} \widehat{\omega}_{i_k} \delta_{\underline{\boldsymbol{x}}_{i_k}}$$

set $\ell = \ell + 1$ and go to step 2)

Else If $\Big(\min\big\{N_{\ell+1}, 2M_{\ell+1}\big\} = 2M_{\ell+1}\Big)$

3) Apply the Recombination Algorithm with $2M_{\ell+1}$-Tree Measure Reduction on $\nu^{\mu}_{m,\ell+1}$ and produce the Carathéodory cubature measure of degree m, $\widehat{\nu}^{\mu}_{m,\ell+1}$ for $\mu_{\ell+1}$

$$\widehat{\nu}^{\mu}_{m,\ell+1} = \sum_{k=1}^{M^{'}_{\ell+1}} \widehat{\omega}_{i_k} \delta_{\underline{\boldsymbol{x}}_{i_k}}$$

set $\ell = \ell + 1$ and go to step 2)

Remark 4.0.12. *In the scenario,* $\Big(\min\big\{N_{\ell+1}, 2M_{\ell+1}\big\} = N_{\ell+1}\Big)$*, we are not able to apply the Recombination Algorithm with* $2M$*-Tree Measure Reduction, since we do not have enough nodes to perform the RHFC tree forest construction. Instead, we proceed as follows: In step* 3) *of the Carathéodory Cubature Algorithm, we form the* $(m, \ell+1)$*-monomial form of* $\nu^{\mu}_{m,\ell+1}$ *:*

$$\gamma^{\mu}_{m,\ell+1} := {\gamma^{\ell+1}_{m}}_{*}(\nu^{\mu}_{m,\ell+1}) = \sum_{i=1}^{N_{\ell+1}} \omega_i \delta_{\gamma^{\ell+1}_{m}(\underline{\boldsymbol{x}}_i)}$$

Similarly to step 1) *of the Recombination Algorithm with* $2M$*-Tree Measure Reduction. Then, we set*

$$\eta_{N_{\ell+1}} := \gamma^{\mu}_{m,\ell+1} = \sum_{i=1}^{N_{\ell+1}} \omega_i \delta_{\gamma^{\ell+1}_{m}(\underline{\boldsymbol{x}}_i)}$$

and we proceed to construct a reduced measure $\eta_{M^{'}_{\ell}} := \widehat{\gamma}^{\mu}_{m,\ell+1}$*. We note that since* $\eta_{N_{\ell+1}}$ *does not contain any information relating to binary trees, we produce a reduced measure, rather than a reduced tree measure as in the Recombination Algorithm. We may achieve this, by setting*

$$A^{(0)} = [\gamma^{\ell+1}_{m}(\underline{\boldsymbol{x}}_1), \ldots, \gamma^{\ell+1}_{m}(\underline{\boldsymbol{x}}_{N_{\ell+1}})] \in \mathbb{R}^{M_{\ell+1} \times N_{\ell+1}}$$

$$\underline{\boldsymbol{\beta}}^{(0)} = [\omega_1, \ldots, \omega_{N_{\ell+1}}]^T \in \mathbb{R}^{N_{\ell+1}}_{+}, \ \sum_{i=1}^{N_{\ell+1}} \omega_i = 1$$

Given that $rank(A^{(0)}) = M^{'}_{\ell+1} \leq \min\{M_{\ell+1}, \ N_{\ell+1}\}$*, we have that* $dim(\mathcal{K}er(A^{(0)})) = N_{\ell+1} - M^{'}_{\ell+1}$ *and hence we proceed to form the* $\mathcal{K}er(A^{(0)})$

$$\Phi^{(0)} = [\underline{\boldsymbol{\phi}}^{(0)}_1, \underline{\boldsymbol{\phi}}^{(0)}_2, \ldots, \underline{\boldsymbol{\phi}}^{(0)}_{N_{\ell+1} - M^{'}_{\ell+1}}] \in \mathbb{R}^{N_{\ell+1} \times M_{\ell+1}}$$

and we can then perform the $(N_{\ell+1} - M'_{\ell+1})$*-Tree Measure Reduction, similarly to the degenerate* M*-Tree Measure Reduction, described in Section 3.3.1.*

The aforementioned algorithm, is clearly not the only method for achieving a polynomial runtime for the generation the Carathéodory cubature measure $\widehat{\nu}^{\mu}_{m,d}$ for μ. A natural alternative method would be to consider forming the product in step 2) as follows

$$\nu^{\mu}_{m,2^{\ell}} = \widehat{\nu}^{\mu}_{m,2^{\ell-1}} \otimes \widehat{\nu}^{\mu}_{m,2^{\ell-1}}$$

Heuristically, we have found the approach to be less efficient than the described algorithm, we attribute this fact to the superior memory requirements. An optimal choice of such an algorithm requires further investigation and is beyond the scope of this dissertation.

As noted in Section 1.3.4, by construction of Carathéodory Cubature Algorithm, the constituent sequence of univariate quadrature formulae in (4.1) should be chosen so as to attain the highest degree of precision with the fewest abscissae. Therefore, Gaussian Quadrature measures, are in this context the best choice of the underlying sequence of univariate quadratures.

Let us now fully examine the cost of the Carathéodory Cubature Algorithm, in the context of a measure μ with densities ρ_i, for which we have Gaussian quadrature measures.

Firstly, we note that in the context of Gaussian quadrature measure, we have

$$N_{\ell+1} := supp(\nu^{\mu}_{m,\ell+1}) = M'_{\ell} \times supp(\nu^{\mu}_{m,1}) = M'_{\ell} \times \left(\lfloor \frac{m}{2} \rfloor + 1 \right)$$

and for any given degree $m \in \mathbb{N}$ and dimension $d \in \mathbb{N}$ there exists a threshold $r = r(m,d) \in [\![1, d-1]\!]$ for which

$$N_{\ell+1} \leq 2M_{\ell+1} \qquad \text{if} \quad \ell \leq r$$

$$N_{\ell+1} \geq 2M_{\ell+1} \qquad \text{if} \quad \ell > r$$

For example, for the d-dimensional Lebesgue measure μ, for which we know Gauss-Legendre quadrature formulae, introduced in Section 1.1.3.1, let us consider the computation of the 'Carathéodory" cubature measure $\widehat{\nu}_{m,d}^{\mu}$ of dimension $d = 18$ and degree $m = 4$. For $1 \leq \ell \leq 17$ we have

$$N_{\ell+1} := supp(\nu_{4,\ell+1}^{\mu}) = M_{\ell}^{'} \times supp(\nu_{4,1}^{\mu}) = M_{\ell}^{'} \times \Big(\lfloor \frac{4}{2} \rfloor + 1\Big)$$
$$M_{\ell+1} := \binom{4+\ell+1}{\ell+1}$$
$$M_{\ell+1}^{'} := card(\widehat{\nu}_{4,\ell+1}^{\mu})$$

Then, for comparison, set $N_{\ell+1}^{*} := card(\nu_{\ell+1}^{\mu})$, i.e. $N_{\ell+1}^{*}$ denotes the cardinality of the Cartesian tensor cubature product measure $\nu_{\ell+1}^{\mu}$ of degree 4 in dimension $\ell + 1$ for $\mu_{\ell+1}$. Now we have

$$N_{\ell+1}^{*} := \Big(\lfloor \frac{4}{2} \rfloor + 1\Big)^{\ell+1}$$

DIM= ℓ	M_ℓ	N_ℓ	$M_\ell^{'}$	N_ℓ^*
2	15	9	9	9
3	35	27	23	27
4	79	69	50	81
5	126	150	96	243
6	210	288	168	729
7	330	504	274	2187
8	495	822	423	6561
9	715	1269	625	19683
10	1001	1875	891	59049
11	1365	2673	1233	177147
12	1820	3699	1664	531441
13	2380	4992	2197	1594323
14	3060	6591	2849	4782969
15	3876	8547	3639	14348907
16	4845	10902	4569	43046721
17	5985	13707	5673	129140163
18	7315	17019	6963	387420489

Table 4.1: N_ℓ^* = Cardinality of Cartesian tensor Gaussian Cubature and $M_\ell^{'}$ = Cardinality of Carathéodory Gaussian Cubature

From the table above, it is clear that the threshold $r = 11$, since

$$N_{\ell+1} > 2M_{\ell+1} \qquad \text{for} \quad \ell > 11$$

In particular, we also note that, for each dimension ℓ, the cardinality of the Carathéodory gaussian cubature measure $M_\ell^{'} := card(\widehat{\nu}_{4,\ell}^{\mu}) < M_\ell$. Empirically, we can confirm that this is often the case for a general degree $d \in \mathbb{N}$ and dimension $m \in \mathbb{N}$, in the context of Gaussian-Legendre cubature measures.

Given that we know that there exists a threshold $r = r(m,d) \in [\![1, d-1]\!]$ for which, we have

$$N_{\ell+1} \leq 2M_{\ell+1} \qquad \text{if} \quad \ell \leq r$$

$$N_{\ell+1} \geq 2M_{\ell+1} \qquad \text{if} \quad \ell > r$$

we will split the cost of the overall Carathéodory Cubature Algorithm into two parts.

For $\ell \leq r$, step 2) of the algorithm requires touching $N_{\ell+1}$ nodes of dimension d and $N_{\ell+1}$ weights, hence it has an associated cost of $N_{\ell+1}(d+1)$. In steps 3) − 4), we perform $(N_{\ell+1} - M_{\ell+1})$-Measure Reduction, which has an associated cost of $C_1(\max\{N_{\ell+1}, M_{\ell+1}\}^3)$, for some $C_1 \in \mathbb{R}_+$.

For $\ell > r$, step 2) has an associated cost of $N_{\ell+1}(d+1)$. Whilst in step 3) we perform the Recombination Algorithm with $2M_{\ell+1}$-Tree Measure Reduction, this has an associated cost of

$$2N_{\ell+1}M_{\ell+1} + log\Big(\frac{N_{\ell+1}}{2M_{\ell+1}}\Big)C_2 M_{\ell+1}^3 \qquad \text{for some} \quad C_2 \in \mathbb{R}_+$$

Hence, the overall theoretical computational complexity of the Carathéodory Cubature Algorithm is given by

$$(d+1)\sum_{\ell=1}^{d-1} N_{\ell+1} + C_1 \sum_{\ell=1}^{r} (\max\{N_{\ell+1}, M_{\ell+1}\}^3) + 2\sum_{\ell=r+1}^{d-1} N_{\ell+1}M_{\ell+1} + C_2 \sum_{\ell=r+1}^{d-1} log\Big(\frac{N_{\ell+1}}{2M_{\ell+1}}\Big) M_{\ell+1}^3$$

For a Gaussian cubature measure, the above expression simplifies to

$$(d+1)\Big(\lfloor \frac{m}{2} \rfloor + 1\Big)\sum_{\ell=1}^{d-1} M'_{\ell+1} + C_1 \sum_{\ell=1}^{r} (\max\{\Big(\lfloor \frac{m}{2} \rfloor + 1\Big) M'_{\ell+1}, M_{\ell+1}\}^3) +$$
$$\big(\lfloor m \rfloor + 2\big)\sum_{\ell=r+1}^{d-1} M'_{\ell+1}M_{\ell+1} + C_2 \sum_{\ell=r+1}^{d-1} log\left(\frac{\big(\lfloor m \rfloor + 2\big)M'_{\ell+1}}{4M_{\ell+1}}\right) M_{\ell+1}^3$$

since $M'_{\ell+1} \leq M_{\ell+1} \leq M_d$, by setting $M := M_d$, we can obtain an upper bound in terms of M, consistent with estimates in the previous chapters

$$C_1^*(d,m)M + C_2^*(d,r,m)M^2 + C_3^*(d,r,m)M^3$$

where

$$C_1^* = (d-1)(d+1)\Big(\lfloor\frac{m}{2}\rfloor + 1\Big)$$

$$C_2^* = (d-r+1)\big(\lfloor m\rfloor + 2\big)$$

$$C_3^* = C_1 r\big(\lfloor m\rfloor + 2\big)^3 + C_2(d-r+1)log\left(\frac{\big(\lfloor m\rfloor + 2\big)M}{4M_{r+1}}\right)$$

Hence, the overall theoretical cost of Carathéodory Cubature Algorithm, is cubic in M.

Chapter 5

Numerical Validation

5.0.5 Introduction

In the forthcoming chapter, we present a comparison of numerical integration results, attained by employing Carathéodory Cubature Algorithm, Smolyak Algorithm and Delayed Smolyak Algorithm for the d-dimensional Lebesgue measure on $[0,1]^d$. We test the integration of a well-known suite of numerical integration functions: Genz integration test suite. This test suite comprises of six families of integrands, commonly employed to investigate the accuracy of cubature measures and interpolation schemes.

To test the Carathéodory Cubature Algorithm, we choose the underlying sequence of univariate quadrature measures to be the sequence of Gauss-Legendre quadratures introduced in Section 1.1.3.1. As noted in Chapter 1.1, they provide the highest precision integration with the fewest number of cubature nodes, in the context of Carathéodory Cubature.

Within the context of Smolyak integration, we shall choose the underlying sequence of quadrature measures to be the nested Clenshaw-Curtis and the nested delayed Clenshaw-Curtis sequence. As noted in Section 1.2.4, nested quadrature sequences are the most economical choice for the Smolyak construction, in that they yield the highest precision, with the fewest number of nodes. Moreover, the choice of Clenshaw-Curtis, as opposed to Gauss-Legendre quadratures, is justified by its stability properties, as the norm of the Smolyak construction is generally much smaller, when the underlying sequence of quadratures is Clenshaw-Curtis, see equation (1.29).

All of the forthcoming numerical experiments were conducted in 64-bit(double precision) an SGI UV 100 Shared Memory System, comprising of 8 Intel Xeon CPU $E7-8837$ processors giving access to 64 cores and 1TB of Random Access Memory. Each of these cores run at a 2.67 GHz clock speed with cache sizes Level 1: 64KB, Level 2: 256KB and a shared Level 3 cache of 24MB per CPU. The operating system was SUSE Linux Enterprise Server 11. We note that not all the CPU and memory resources were utilised during our numerical experiments, most experiments required 32 CPUs and up to 400 GB of shared memory. The Carathéodory Cubature Algorithm is a proprietary software package and the Smolyak and Delayed Smolyak Algorithms is also a proprietary software package, available at [115].

Carathéodory Cubature Algorithm software package is a joint effort of the author, Prof. T. Lyons, Dr. C. Litterer, Dr. N. Victoir and Prof. M. Giles. The development of this algorithm is tiered. The first tier, consists of the Recombination Algorithm with 1-Tree Measure Reduction, algorithm 4 in Chapter 3, referenced in [96], was developed by Prof. T. Lyons incorporating contributions from Dr. C. Litterer and Dr. N. Victoir. The second tier of development of this algorithm consists of the Updating version of Recombination Algorithm with 1-Tree Measure Reduction, described in Section 3.2.0.12, as suggested by Prof. M. Giles and developed by the author. The third tier was the development of the Recombination Algorithm with M-Tree Measure Reduction, which is the final version of the Recombination Algorithm, integrated into the Carathéodory Cubature Algorithm. Both Recombination Algorithm with M-Tree Measure Reduction and Carathéodory Cubature Algorithm were developed jointly by Prof. T Lyons and the author.

All the following numerical comparisons on Genz Test Suite for Carathéodory Cubature Algorithm and Smolyak, Delayed Smolyak Cubature Methods were developed by the author, utilising the aforementioned software packages.

5.0.6 Testing Procedures

The functions comprising Genz integration test suite, with parameters $\underline{\boldsymbol{c}} = (c_1, \ldots, c_d)$, $\underline{\boldsymbol{\omega}} = (\omega_1, \ldots, \omega_d)$, defined on $[0,1]^d$, are given by

1) OSCILLATORY $\quad f_1(x) = cos\Big(2\pi\omega_1 + \sum_{i=1}^{d} c_i x_i\Big)$

2) PRODUCT PEAK $\quad f_2(x) = \prod_{i=1}^{d} \dfrac{1}{c_i^2 + (x_i - \omega_i)^2}$

3) CORNER PEAK $\quad f_3(x) = \dfrac{1}{\Big(1 + \sum_{i=1}^{d} c_i x_i\Big)^{d+1}}$

4) GAUSSIAN $\quad f_4(x) = exp\Big(-\sum_{i=1}^{d} c_i^2 (x_i - \omega_i)^2\Big)$

5) CONTINUOUS $\quad f_5(x) = exp\Big(-\sum_{i=1}^{d} c_i |x_i - \omega_i|\Big)$

6) DISCONTINUOUS $\quad f_6(x) = \begin{cases} 0 & \text{if} \quad x_1 > \omega_1 \quad \text{and} \quad x_2 > \omega_2 \\ exp\Big(\sum_{i=1}^{d} c_i x_i\Big) & \text{otherwise} \end{cases}$

Different test integrals can obtained by varying the parameters $\underline{\boldsymbol{c}} = (c_1, \ldots, c_d)$ and $\underline{\boldsymbol{\omega}} = (\omega_1, \ldots, \omega_d)$. Given a pre-determined sample size S, it is common to generate S realisations of each given Genz test function in a given dimension, by randomly generating S suitable pairs $\underline{\boldsymbol{c}}$ and $\underline{\boldsymbol{w}}$. The vector $\underline{\boldsymbol{w}}$ acts as a shift parameter, so its components are chosen to be uniformly distributed in $[0,1]$. The difficulty of each problem is controlled by the vector $\underline{\boldsymbol{c}}$, with the difficulty of the integral usually increasing as the Euclidean norm $||\underline{\boldsymbol{c}}||$ is increased, see [107].

In the numerical experiments presented in this section, we will consider sample sizes of $S = 20$ for each Genz function f_j, $1 \leq j \leq 6$ in dimensions $2, 3, 4$ and the difficulty of each integrand function will be controlled by $||\underline{\boldsymbol{c}}||_1 = b_j$, where b_j is chosen as follows

j	1	2	3	4	5	6
b_j	9.0	7.25	1.85	7.03	1.27	2.0

Table 5.1: Genz Function Suite Parameters

Thus we generate 20 vectors $\underline{\boldsymbol{w}}$ and $\underline{\boldsymbol{c}}$, independently and uniformly distributed on $[0,1]^d$, with $\underline{\boldsymbol{c}}$ re-normalised, such that its l-1 norm satisfies the aforementioned criteria and hence obtain 20 integrands $\{f_j^s\}_{s=1}^{20}$. For each family $\{f_j^s\}_{s=1}^{20}$, $1 \leq j \leq 6$, dimension $2, 3, 4$ we compute the average number of correct digits in the numerical integral approximation for

$$\frac{\sum_{s=1}^{20} -log_{10}\left(\left|I_d(f_j^s) - Q^{SM}(\ell, d)[f_j^s]\right|\right)}{20}$$

$$\frac{\sum_{s=1}^{20} -log_{10}\left(\left|I_d(f_j^s) - Q^{SMD}(p, d)[f_j^s]\right|\right)}{20}$$

$$\frac{\sum_{s=1}^{20} -log_{10}\left(\left|I_d(f_j^s) - Q^{CAR}(m, d)[f_j^s]\right|\right)}{20}$$

where the exact integrals $I_d(f_j)$ are known, see for example [114] and as we recall from Chapter 1: $Q^{SM}(\ell, d)$ denotes the ℓ^{th} level Smolyak cubature, $Q^{SMD}(p, d)$ denotes the p^{th} level Delayed Smolyak cubature and $Q^{CAR}(m, d)$ denotes the Carathéodory cubature of degree m.

In each of the following simulations, we compute $Q^{CAR}(m,d)$ for degrees $m = 2,4,6,8,\ 10,12,14,16,18,20$.

We know from lemma (1.2.10) that $deg(Q^{SM}(\frac{m-1}{2},d)) \geq m$ and by construction $deg(Q^{SMD}(\frac{m-1}{2},d)) > deg(Q^{SM}(\frac{m-1}{2},d))$. However, we do not attempt to compare the three integration methods by degree of precision, as this comparison usually yields the cardinality of the Smolyak cubature $card\Big(Q^{SM}(\frac{m-1}{2},d)\Big)$ to be several orders of magnitude greater than $card\Big(Q^{CAR}(m,d)\Big)$ or $card\Big(Q^{SMD}(\frac{m-1}{2},d)\Big)$. Instead, we employ fixed degrees of precision for $Q^{CAR}(m,d)$, $m = 2,4,\ldots,20$ in dimension $d = 2,3,4$.

We employ varying levels ℓ, p for $Q^{SM}(\ell,d)$ and $Q^{SMD}(p,d)$ thus varying degrees of precision, whilst yielding a comparable number of nodes to $Q^{CAR}(m,d)$. As such, we study the accuracy of each method as a function of the underlying cardinality of cubature nodes.

In dimension $d = 2$, we employ $Q^{SM}(\ell,2)$ with levels $\ell = 1,2,\ldots,6$ corresponding to degrees $m = 3,5,\ldots,13$; we employ $Q^{SMD}(p,2)$ with levels $p = 1,2,\ldots,9$ corresponding to degrees $m = 3,5,\ldots,19$.

In dimension $d = 3$, we employ $Q^{SM}(\ell,3)$ with levels $\ell = 1,2,\ldots,8$ corresponding to degrees $m = 3,5,\ldots,17$; we employ $Q^{SMD}(p,3)$ with levels $p = 1,2,\ldots,12$ corresponding to degrees $m = 3,5,\ldots,25$.

In dimension $d = 4$, we employ $Q^{SM}(\ell,4)$ with levels $\ell = 1,2,\ldots,8$ corresponding to degrees $m = 3,5,\ldots,17$; we employ $Q^{SMD}(p,4)$ with levels $p = 1,2,\ldots,14$ corresponding to degrees $m = 3,5,\ldots,29$.

The levels, corresponding degrees and cardinalities of $Q^{SM}(\ell,d)$ and $Q^{SMD}(p,d)$ for each dimension $d = 2,3,4$ are illustrated below. We also provide the cardinalities of $Q^{CAR}(m,d)$ for degrees $m = 2,4,\ldots,20$.

degree m	$card\Big(Q^{CAR}(m,2)\Big)$	$card\Big(Q^{CAR}(m,3)\Big)$	$card\Big(Q^{CAR}(m,4)\Big)$
2	4	8	15
4	9	27	70
6	16	64	210
8	25	125	494
10	36	216	1000
12	49	343	1820
14	64	512	3060
16	81	729	4844
18	100	1001	7315
20	121	1331	10626

Table 5.2: Cardinalities of $Q^{CAR}(m,d)$ for dimensions $d = 2,3,4$, degree $m = 2,4,\ldots,20$

level ℓ	*degree* m	$card\Big(Q^{SM}(\ell,2)\Big)$	$card\Big(Q^{SMD}(\ell,2)\Big)$
1	3	5	5
2	5	13	9
3	7	29	17
4	9	65	33
5	11	145	33
6	13	321	65
7	15		97
8	17		97
9	19		161

Table 5.3: Cardinalities of $Q^{SM}(\ell,2)$ and $Q^{SMD}(\ell,2)$ in dimension $d = 2$

level ℓ	*degree* m	$card\Big(Q^{SM}(\ell,3)\Big)$	$card\Big(Q^{SMD}(\ell,3)\Big)$
1	3	7	7
2	5	25	19
3	7	69	39
4	9	177	87
5	11	441	135
6	13	1073	207
7	15	2561	399
8	17	6017	495
9	19		751
10	21		1135
11	23		1135
12	25		1759

Table 5.4: Cardinalities of $Q^{SM}(\ell,3)$ and $Q^{SMD}(\ell,3)$ in dimension $d=3$

$level\ \ell$	$degree\ m$	$card\Big(Q^{SM}(\ell,4)\Big)$	$card\Big(Q^{SMD}(\ell,4)\Big)$
1	3	9	9
2	5	41	33
3	7	137	81
4	9	401	193
5	11	1105	385
6	13	2929	641
7	15	7537	1217
8	17	18945	1985
9	19		2881
10	21		4929
11	23		6465
12	25		8705
13	27		13697
14	29		16001

Table 5.5: Cardinalities of $Q^{SM}(\ell,4)$ and $Q^{SMD}(\ell,4)$ in dimension $d=4$

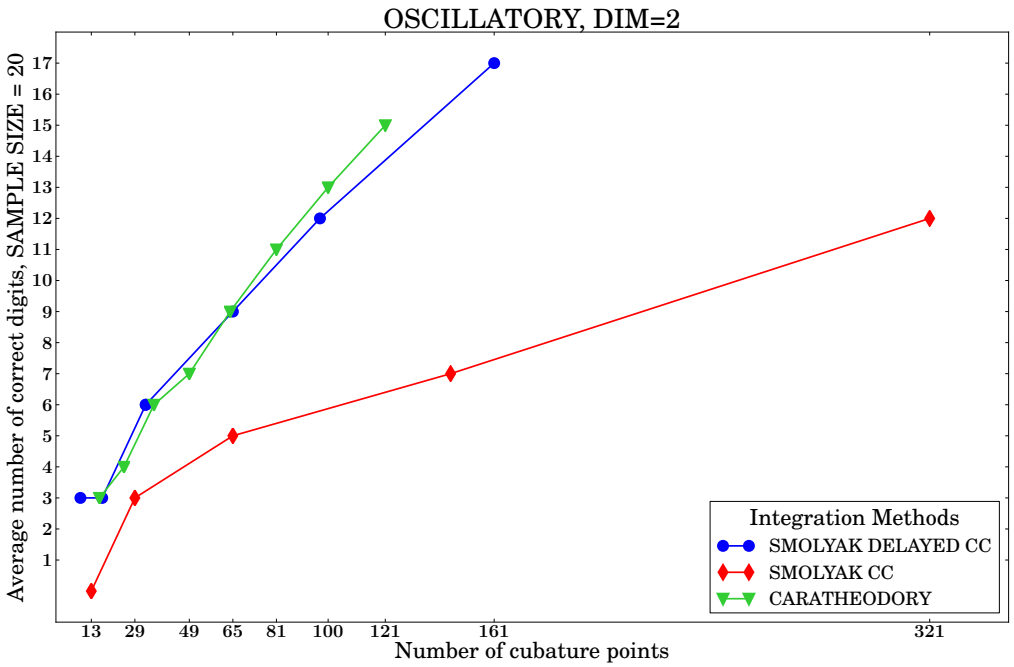


Figure 5.1: Oscillatory, Dim=2

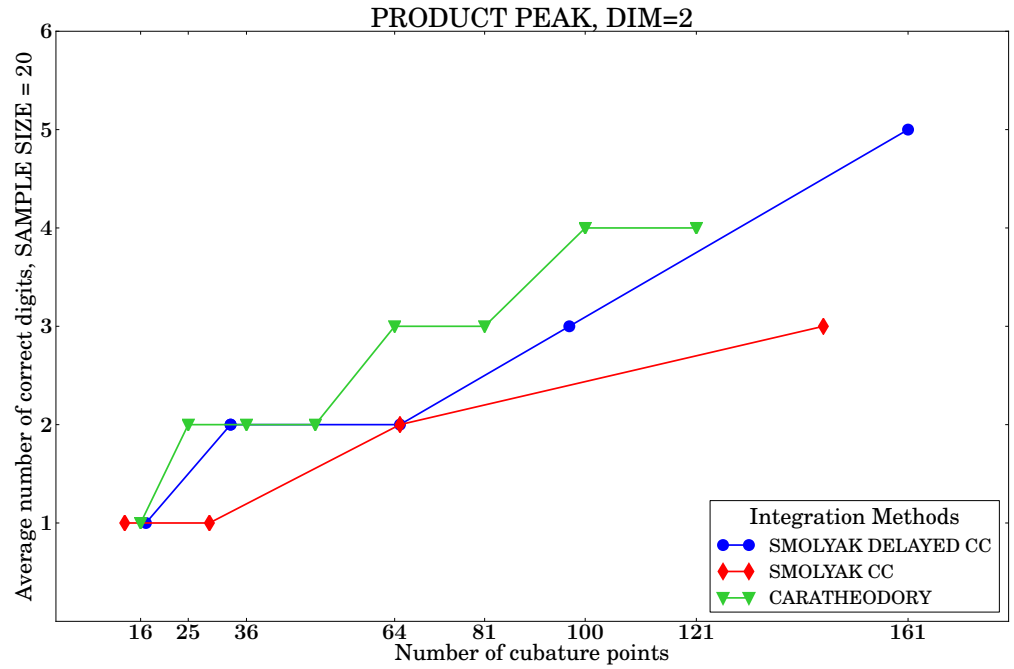


Figure 5.2: Product Peak, Dim=2

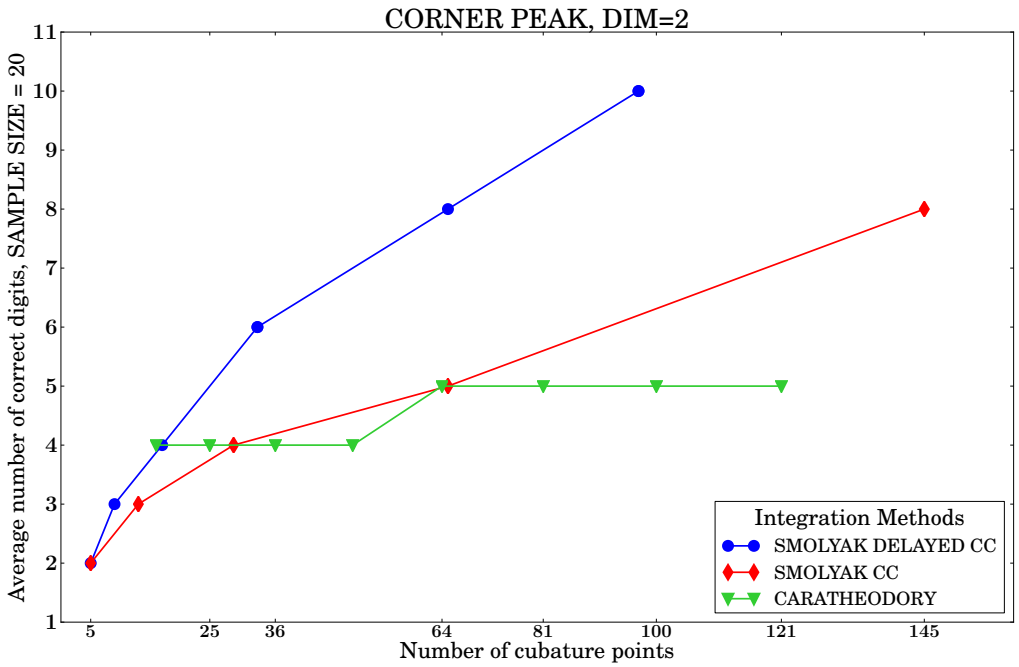


Figure 5.3: Corner Peak, Dim=2

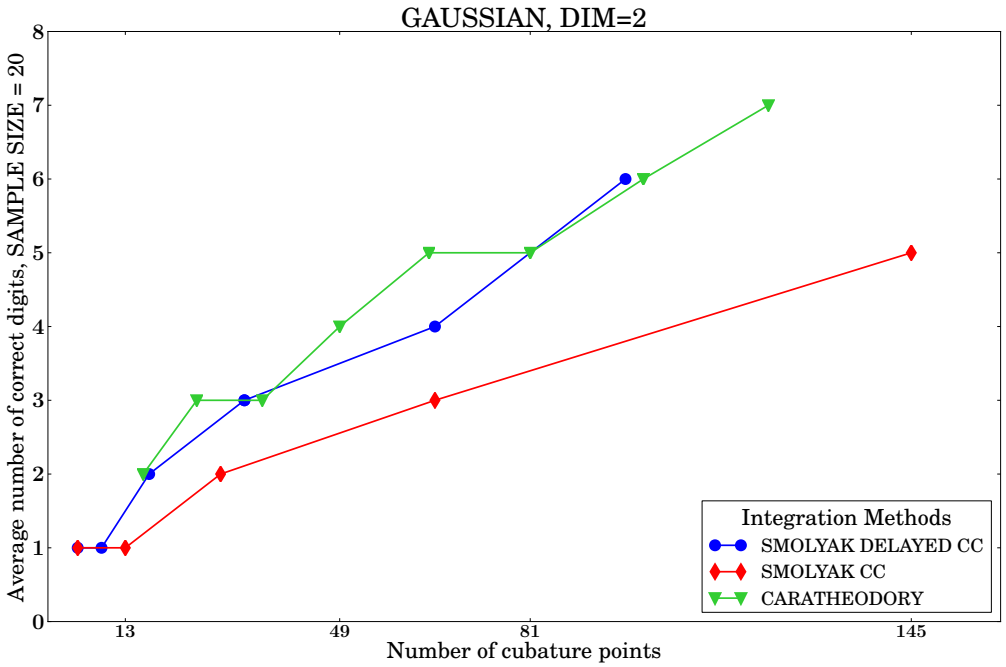


Figure 5.4: Gaussian, Dim=2

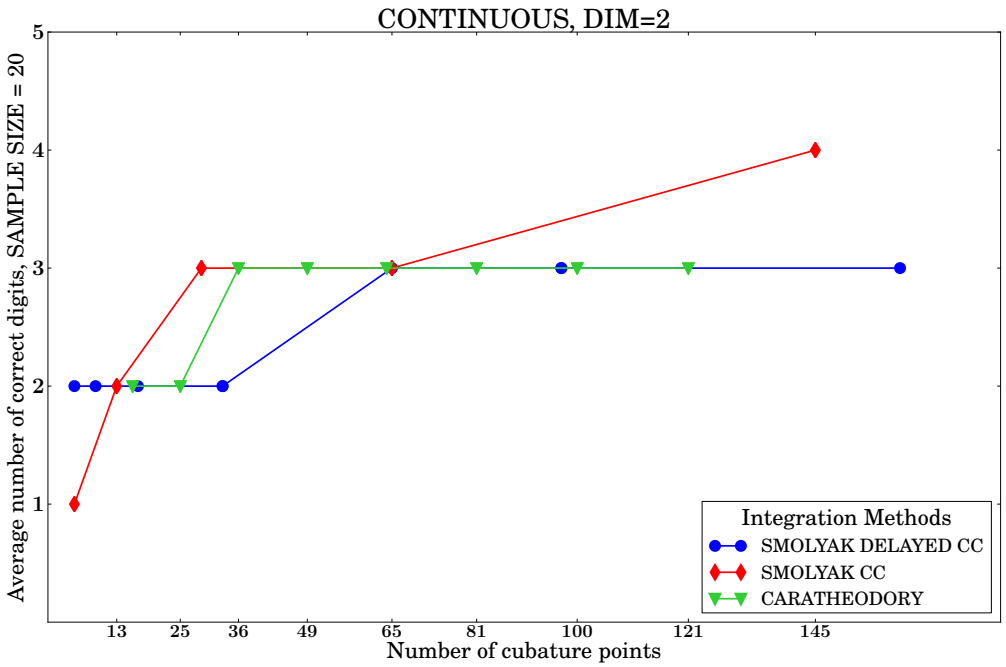


Figure 5.5: Continous, Dim=2

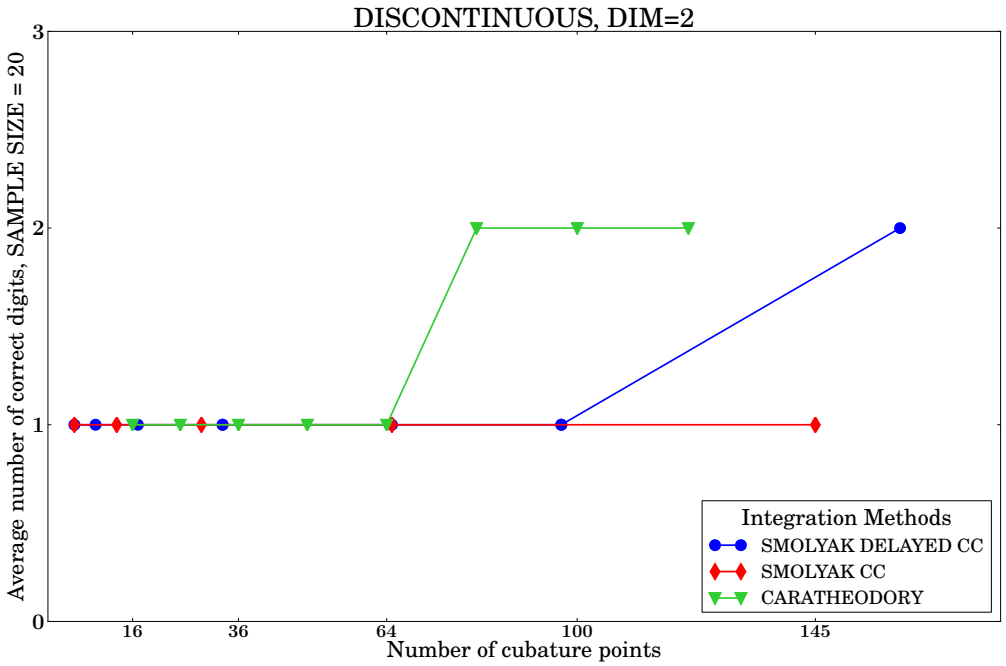


Figure 5.6: Discontinuous, Dim=2

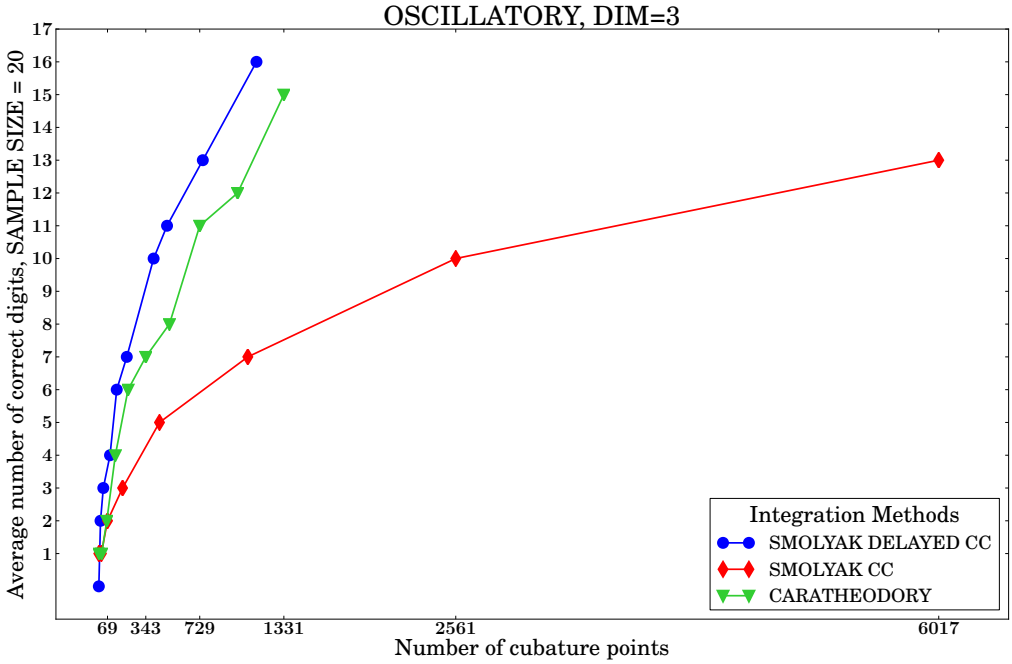


Figure 5.7: Oscillatory, Dim=3

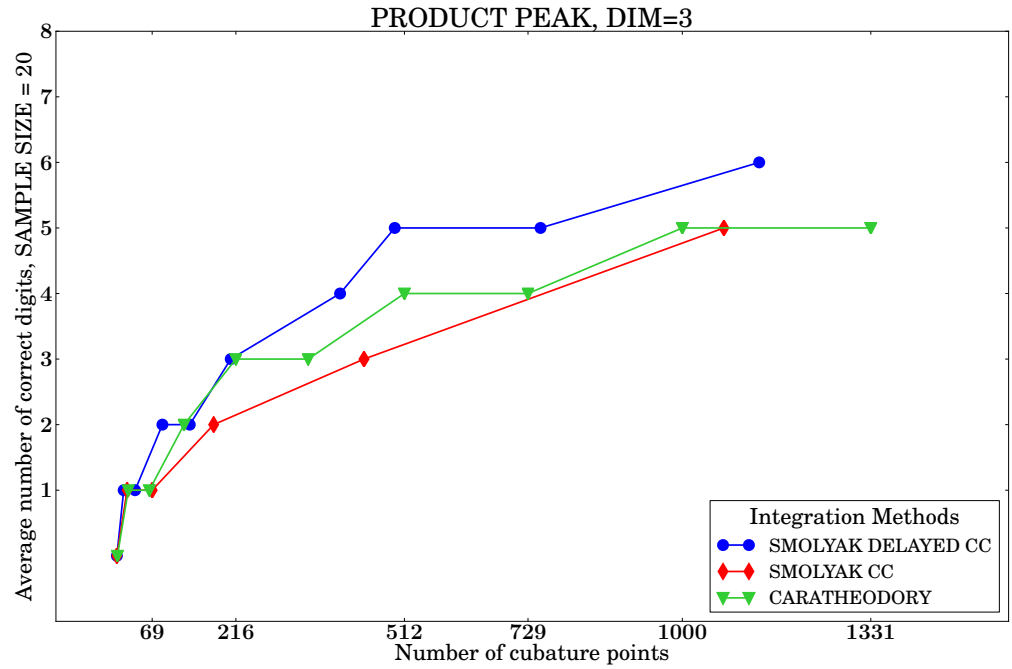


Figure 5.8: Product Peak, Dim=3

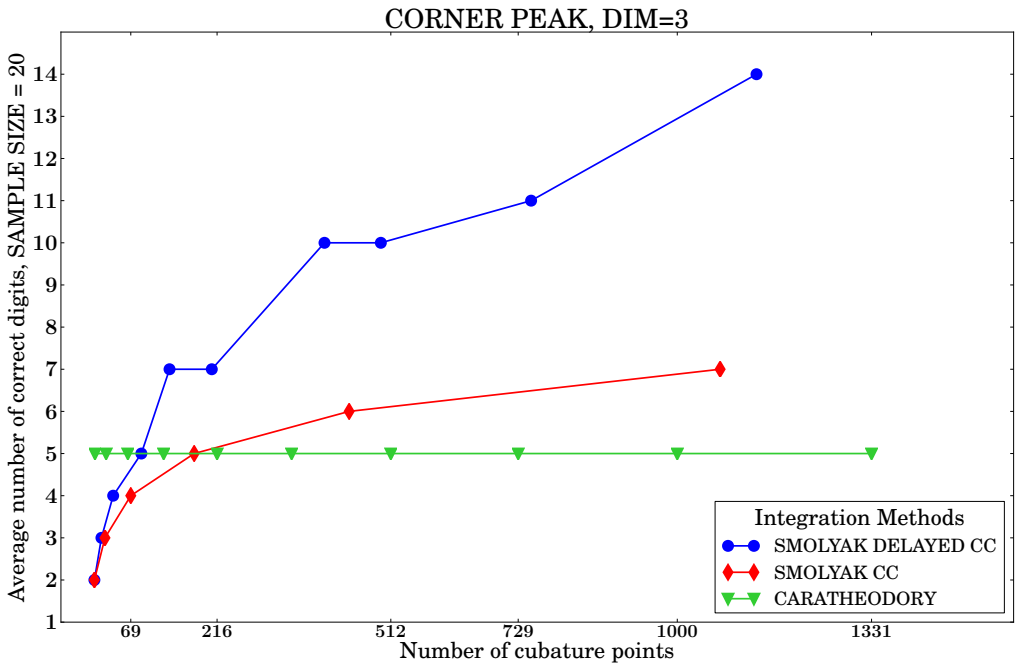


Figure 5.9: Corner Peak, Dim=3

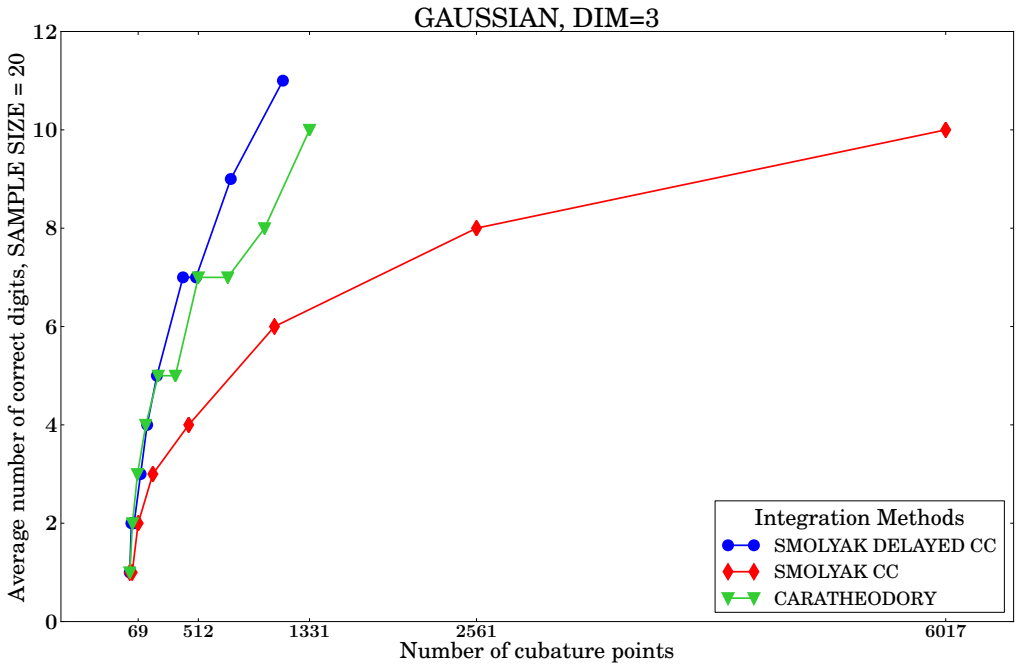


Figure 5.10: Gaussian, Dim=3

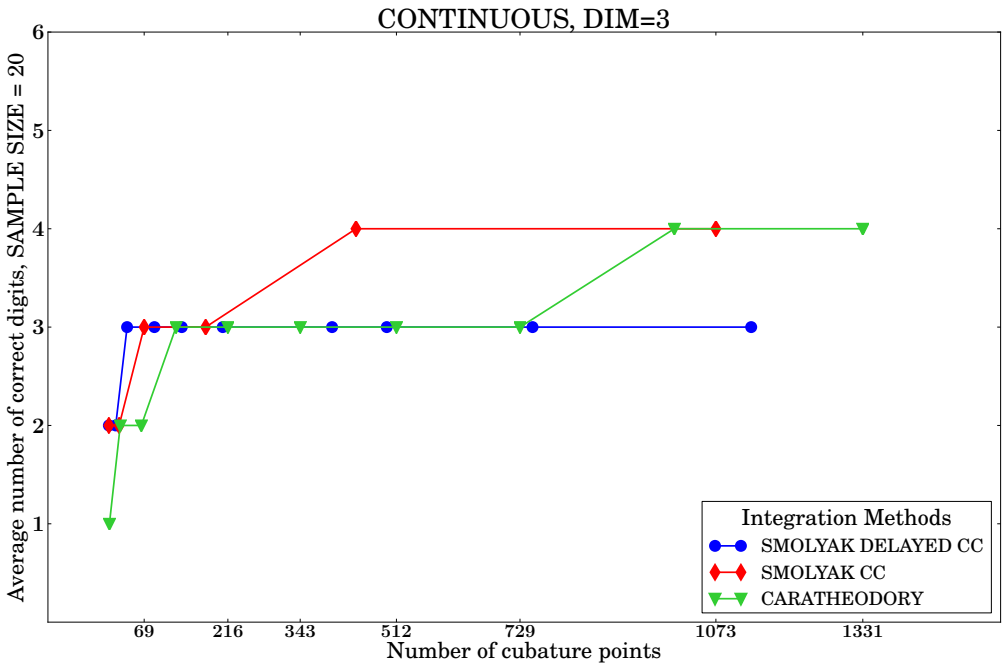


Figure 5.11: Continous, Dim=3

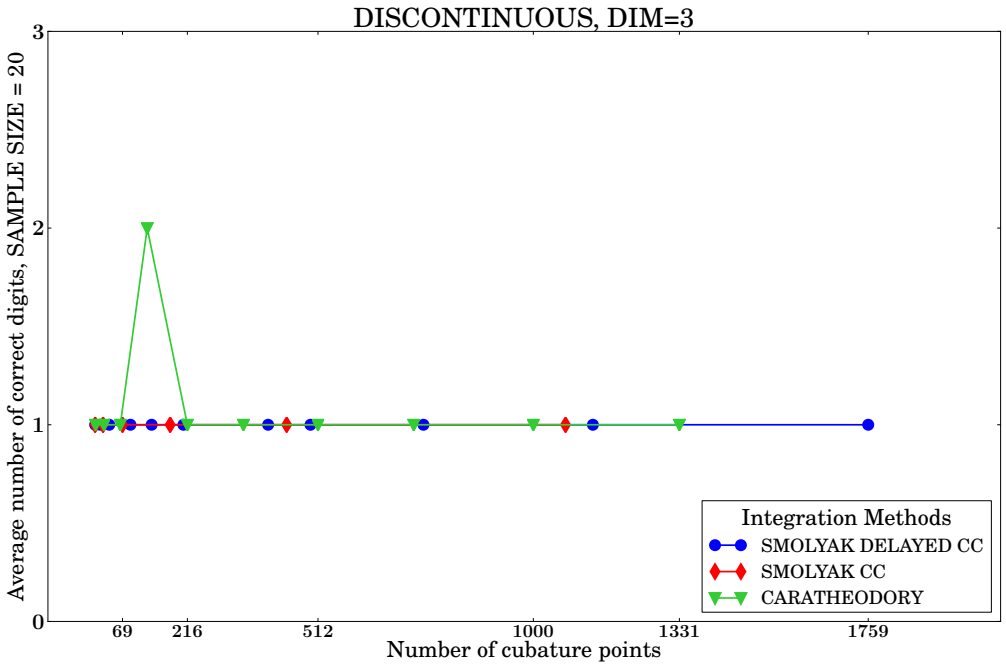


Figure 5.12: Discontinuous, Dim=3

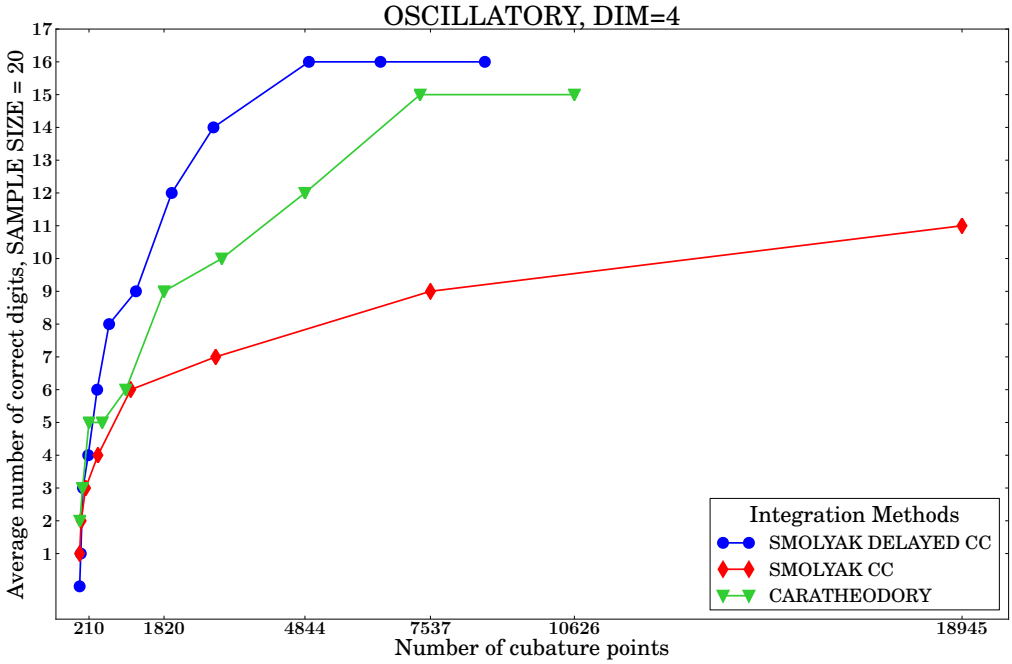


Figure 5.13: Oscillatory, Dim=4

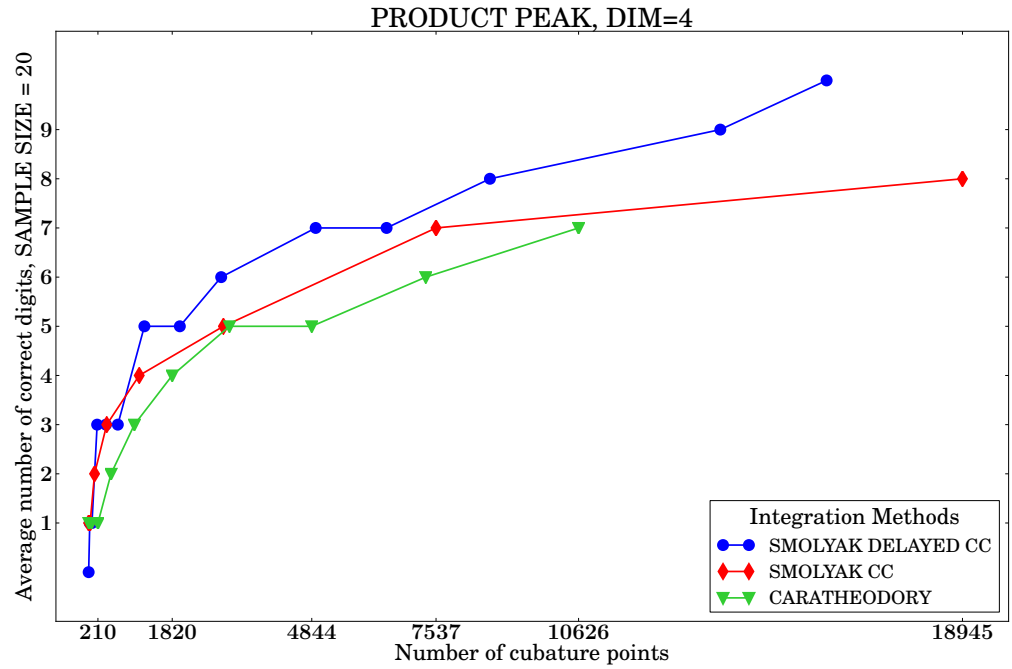


Figure 5.14: Product Peak, Dim=4

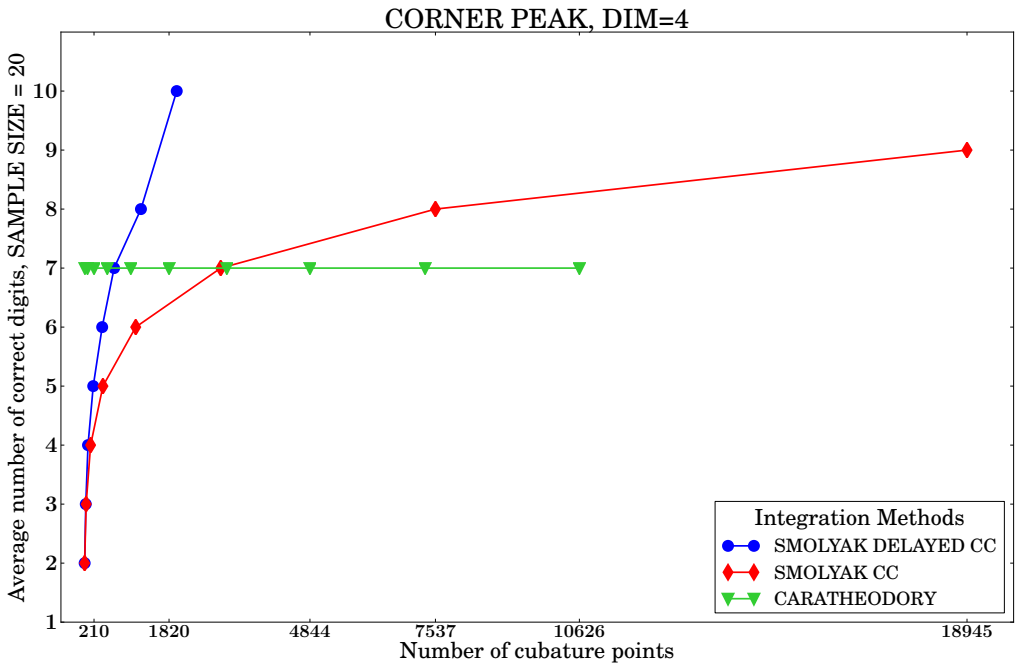


Figure 5.15: Corner Peak, Dim=4

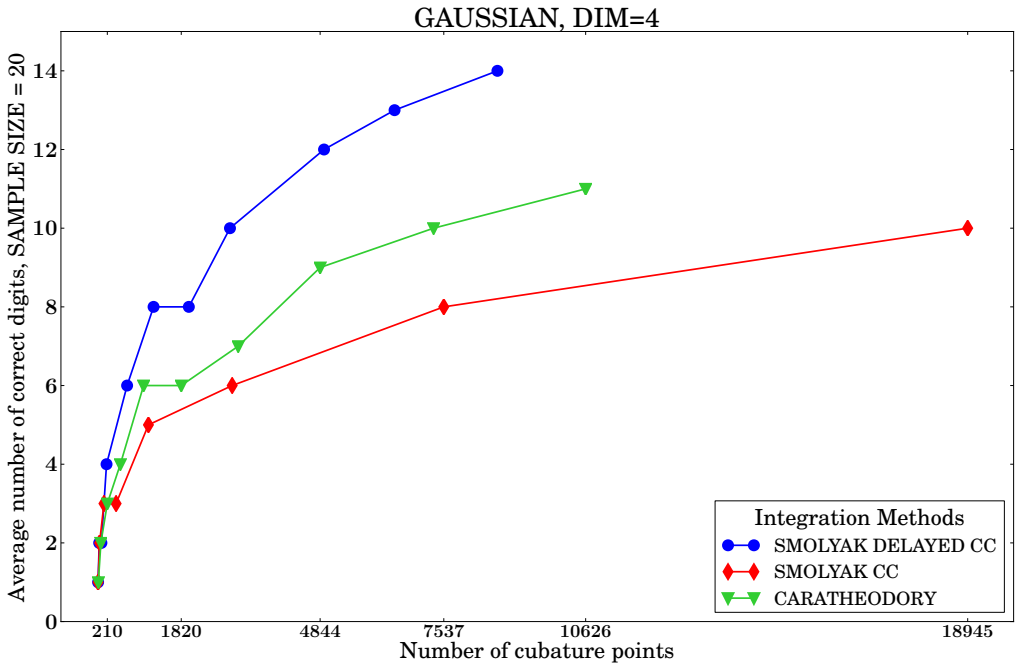


Figure 5.16: Gaussian, Dim=4

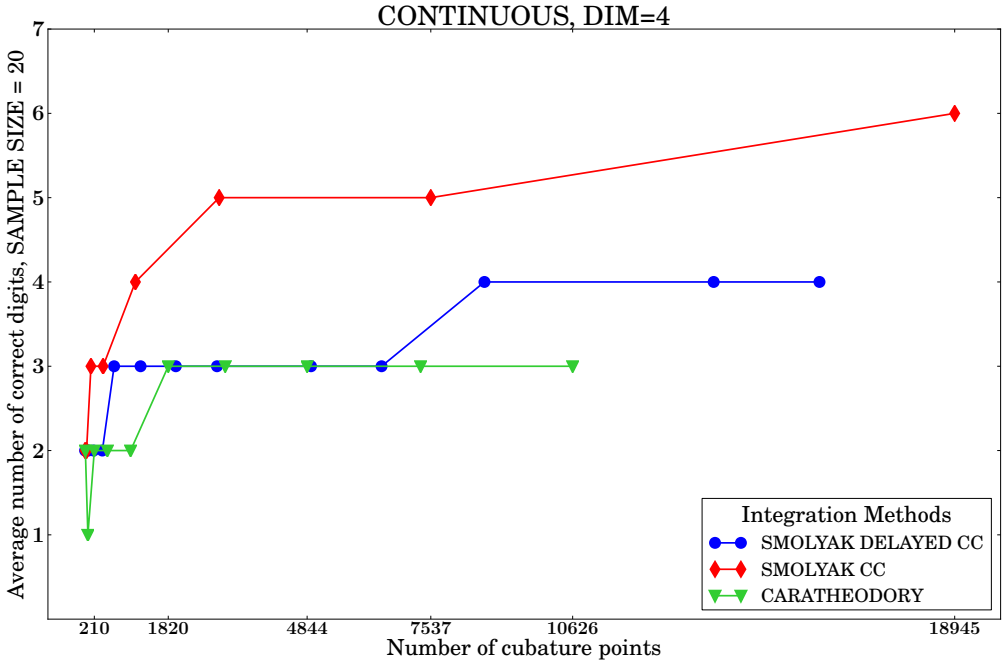


Figure 5.17: Continous, Dim=4

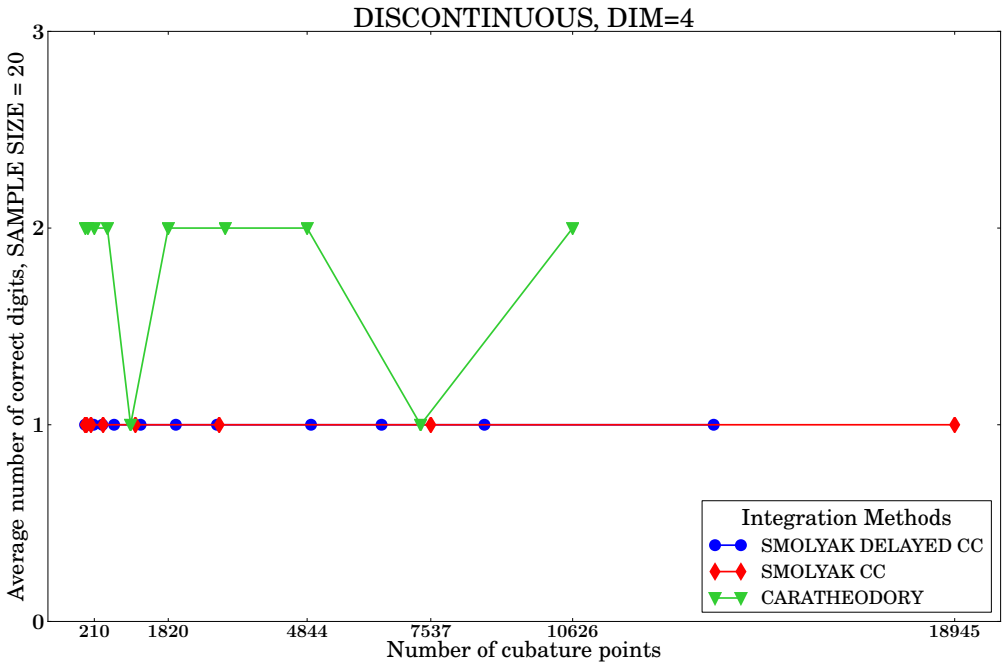


Figure 5.18: Discontinuous, Dim=4

From the above numerical experiments some simple conclusions can be drawn:

- For the Oscillatory and Gaussian Genz functions, the ordinary Smolyak Cubature construction is sharply outperformed by both the Carath?eodory and Delayed Smolyak Cubature methods
- Apart from the Corner Peak and Continuous functions, the Carathéodory and Delayed Smolyak Cubature methods generally yield twice the accuracy for the same number of nodes as the Smolyak Cubature method, independent of dimension
- In dimension 2 the precision of Carathéodory Cubature is higher than that of the Delayed Smolyak Cubature, for all functions, apart from the Corner Peak. Whilst in dimension 3 and 4 the Delayed Smolyak cubature generally yields higher precision than Carathéodory Cubature .
- The performance for Carathéodory Cubature, for Corner Peak, is flat, that is non-increasing with increasing degrees of precision of the method. This type of function may benefit from utilising Carathéodory Cubature adaptively to zoom in on the corner singularity.
- The only function for which the ordinary Smolayk cubature outperforms the other two integration methods, in every dimension, is the continuous function. This may benefit from a greater number of cubature nodes in integrating around its peak
- All three methods appear to be ineffective for the discontinuous function, although Carathéodory Cubature yields the best performance. This may be the result of insufficient number of points around the discontinuity to resolve the jump.

In addition, we note that whilst Carathéodory Cubature measures for all degrees $1 \leq m \leq 20$ contain strictly positive weights, the Smolyak Cubature and the Delayed Smolyak Cubature for each level contain negative weights. Below, we provide the number of negative weights for the Smolyak Cubature and the Delayed Smolyak Cubature for each level and each dimension employed.

ℓ	$card\Big(Q^{SM}(\ell,2)\Big)$	$card\Big(Q^{SMD}(\ell,2)\Big)$	N.W. IN $Q^{SM}(\ell,2)$	N.W. IN $Q^{SMD}(\ell,2)$
1	5	5	1	1
2	13	9	2	1
3	29	17	3	2
4	65	33	5	2
5	145	33	9	2
6	321	65	18	4

Table 5.6: Negative Weights in $Q^{SM}(\ell,d)$ and $Q^{SMD}(\ell,d)$ in dimension $d=2$, level ℓ

ℓ	$card\Big(Q^{SM}(\ell,3)\Big)$	$card\Big(Q^{SMD}(\ell,3)\Big)$	N.W. IN $Q^{SM}(\ell,3)$	N.W. IN $Q^{SMD}(\ell,3)$
1	7	7	1	0
2	25	19	2	1
3	69	39	4	2
4	177	87	8	4
5	441	135	15	4
6	1073	207	29	7
7	2561	399	66	10
8	6017	495	141	10

Table 5.7: Negative Weights in $Q^{SM}(\ell,d)$ and $Q^{SMD}(\ell,d)$ in dimension $d=3$, level ℓ

ℓ	$card\Big(Q^{SM}(\ell,4)\Big)$	$card\Big(Q^{SMD}(\ell,4)\Big)$	N.W. IN $Q^{SM}(\ell,4)$	N.W. IN $Q^{SMD}(\ell,4)$
1	9	9	1	1
2	41	33	2	1
3	137	81	4	3
4	401	193	8	4
5	1105	385	13	4
6	2929	641	30	7
7	7537	1217	67	14
8	18945	1985	145	15

Table 5.8: Negative Weights in $Q^{SM}(\ell,d)$ and $Q^{SMD}(\ell,d)$ in dimension $d=4$, level ℓ

5.0.7 Comparison of Carathéodory Cubature Algorithm Implementations with Dual Simplex and Interior Point Method and SVD Solver

Building on the ideas introduced in Section 1.4, in this section we report the performance attained by Carathéodory Cubature Algorithm with different measure measure reduction implementations: Gurobi Dual Simplex LP Solver, IBM CPLEX Dual Simplex LP Solver and Intel MKL SVD Solver. As in the previous section the Carathéodory Cubature Algorithm is applied to the d-dimensional Lebesgue measure on $[0,1]^d$ with the underlying sequence of Gauss-Legendre univariate quadratures. In this numerical experiment, however we record a series of performance metrics for the different implementations of the Carathéodory Cubature Algorithm, thus highlighting the respective merits and limitations of each implementation.

The versions of underlying LP Solvers utilised are contained in state-of-the-art software packages: Gurobi Optimizer version 6.5 C++ Interface, IBM CPLEX Optimization Studio 12.6.2 C++ Interface and proprietary software package utilising Intel Math Kernel Library 11.1.1 C Interface, shipped with Intel Composer XE Edition 2013 SP1.

We note that the following comparison is performed in a single threaded environment as both Gurobi Dual Simplex LP Solver and IBM CPLEX Dual Simplex LP Solver are only implemented to run on a single thread. As noted in Section 1.3.1, this limitation is intrinsic to the underlying Primal and Dual Simplex Algorithms. However, as noted in the documentation for Gurobi Optimizer version 6.5, IBM CPLEX Optimization Studio 12.6.2 , the Dual Simplex has a smaller memory footprint when compared to the Primal Simplex implementation and thus is preferable to our type of problem. The following numerical experiments will compare runtime, floating point operation count, memory and scalability of the aforementioned solvers, these were implemented by the author.

We utilise the following performance metrics, as means of comparison between the different measure reduction implementations: Wall-Clock Time, MFLOPs (Millions of Floating Point Operations), Memory Footprint Resident and Virtual. In order to obtain accurate performance comparisons we employ PAPI(Performance Application Programming Interface) [116] , which supports measurements of the aforementioned performance metrics from each thread spawned by the application and collecting hardware performance counter values.

In particular, we obtain the aforementioned performance metrics by collecting individual application thread statistics for N total threads and computing

- Wall-Clock Time= $\sum_{i=1}^{N} \frac{\text{(Wall Clock Tick Count per thread } i \text{)}}{\text{Clock Rate}}$ in ticks
  where Clock Rate= 2.67 GHz
- MFLOPS= $\sum_{i=1}^{N}$(PAPI FP OPS per thread i)
  where PAPI FP OPS measures CPU floating point instructions on thread i
- Memory Resident = $\sum_{i=1}^{N}$ (Memory Resident on thread i) in MB
- Memory Virtual = $\sum_{i=1}^{N}$ (Memory Virtual on thread i) in MB

The data sample in this numerical experiment consists of performing Carathéodory Cubature Algorithm for degree $m = 4$ in increasing dimensions $d = 3, 4, \ldots, 30$.

dim= d	$M = \binom{m+d}{d}$	dim= d	$M = \binom{m+d}{d}$
3	35	17	5985
4	70	18	7315
5	126	19	8855
6	210	20	10626
7	330	21	12650
8	495	22	14950
9	715	23	17550
10	1001	24	20475
11	1365	25	23751
12	1820	26	27405
13	2380	27	31465
14	3060	28	35960
15	3876	29	40920
16	4845	30	46376

Table 5.9: Problem size: $M = \binom{m+d}{d}$ for degree $m = 4$, dimensions $d = 3, 4, \ldots, 30$

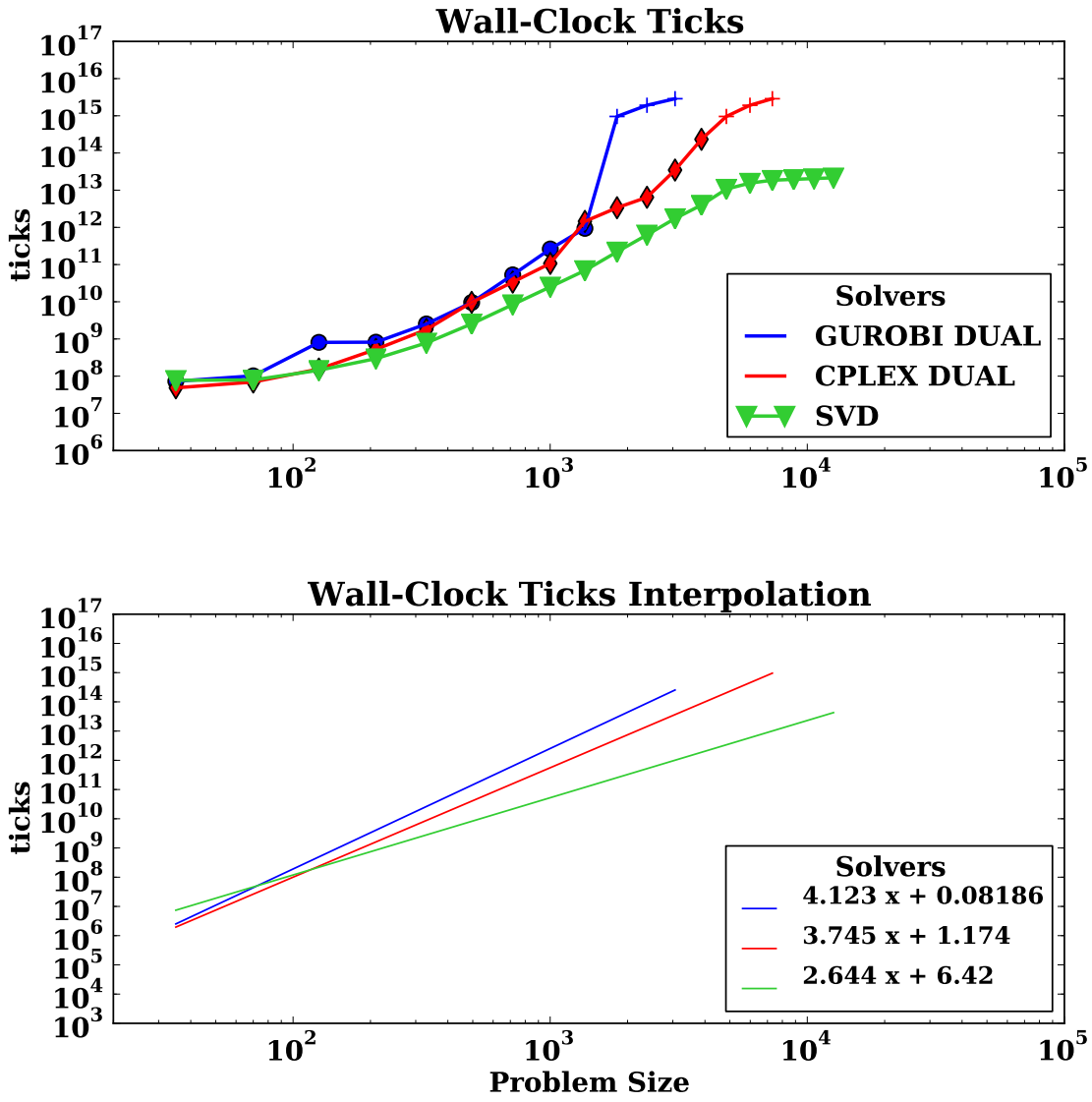


Figure 5.19: Log-log plot of Wall-Clock ticks: Gurobi Dual Simplex, IBM CPLEX Dual Simplex, SVD

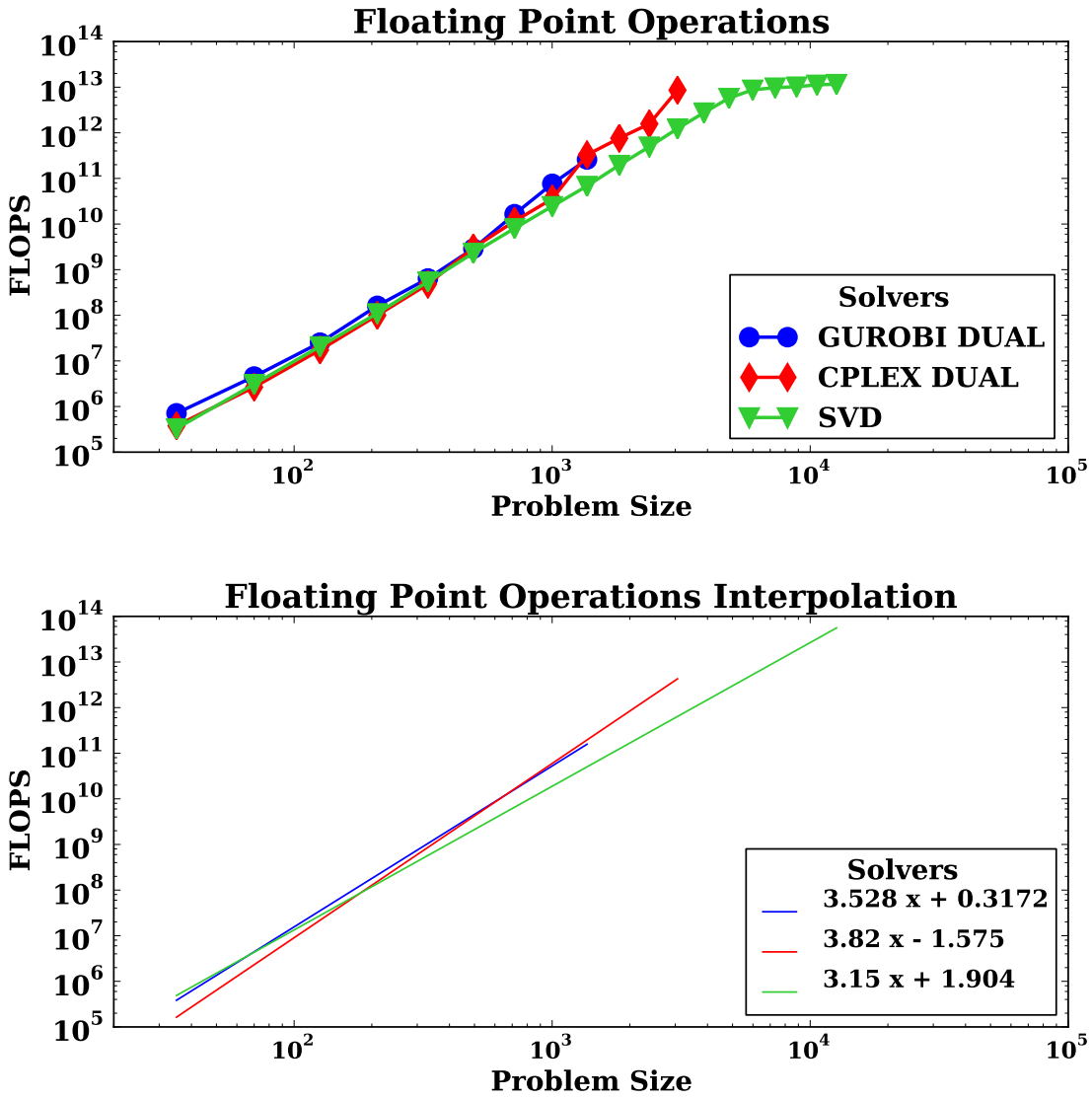


Figure 5.20: Log-log plot of Floating Point Operations: Gurobi Dual Simplex, IBM CPLEX Dual Simplex, SVD

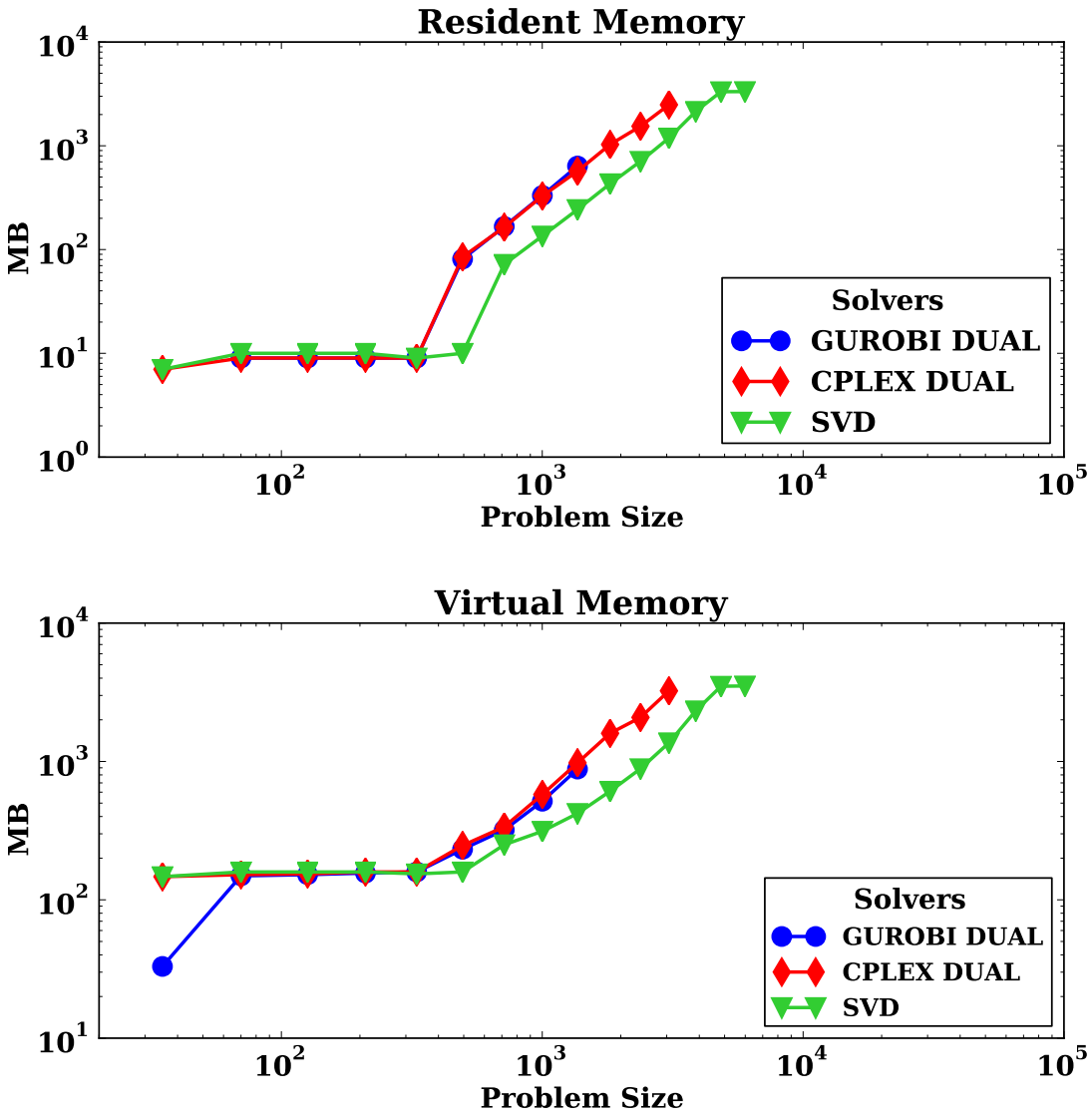


Figure 5.21: Resident and Virtual Memory: Gurobi Dual Simplex, IBM CPLEX Dual Simplex, SVD

We note that both Gurobi Dual Simplex LP Solver and IBM CPLEX Dual Simplex LP Solver were stopped during execution, as the computation for dimension $d = 10$ for Gurobi LP Solver and for dimension $d = 13$ were taking longer than 100 hours to execute. Thus, in the Wall-clock ticks figure (5.34), we employ the cross signs to indicate the lower bound on the number ticks that would result from the computation if each had finished performing in $100, 200, 300$ hours respectively. We note that both Wall-clock ticks in figure (5.34) and Floating Point Operations count in figure (5.20) are plotted on a logarithmic scale on both axes. Thus, power relationships of the form $C(M) = aM^k$ will appear as straight lines on the log-log plots. To estimate the power k corresponding to the slope of the line on the log-log plots we employ linear interpolation. Hence, we can conclude that the empirical runtime complexity for the three implementations of Carathéodory Cubature Algorithm are given by

$$C_{GUR\ DUAL}\Big(Q^{CAR}(m,d)\Big) = O(M^{4.123})$$
$$C_{CPL\ DUAL}\Big(Q^{CAR}(m,d)\Big) = O(M^{3.745})$$
$$C_{SVD}\Big(Q^{CAR}(m,d)\Big) = O(M^{2.644})$$

As noted in Chapter 4, the theoretical cost of the Carathéodory Cubature Algorithm is dominated by the runtime cost of computing the measure reduction. Thus, we conclude that the computational complexity of performing the measure reduction, via Gurobi Dual Simplex LP Solver, IBM CPLEX Dual Simplex LP Solver and SVD can be approximated by $C_{GUR\ DUAL}\big(M, L\big) \approx O(M^{4.123})$, $C_{CPL\ DUAL}\big(M, L\big) \approx O(M^{3.745})$, $C_{SVD}\big(M, L\big) \approx O(M^{2.644})$, where $L = \min\{N, M\}$ as detailed in Chapter 4. Similar conclusions can be drawn for the Floating Point Operations count.

Additionally, from figures (5.20) and (5.21) we note that whilst the number Floating Point Operations employed by each implementation is very similar, the memory consumption in terms of both resident and virtual memory is smaller for the SVD-based implementation. Thus, overall we can conclude that the SVD-based implementation manages memory more efficiently, than the other two methods, which may be a contributing factor to a faster execution.

In the following we present a comparison of accuracy of the three implementations of the Carathéodory Cubature Algorithm. To do so, we perform the same set of simulations used to compare the accuracy of Carathéodory Cubature Algorithm with the Smolayk and the Delayed Smolyak Algorithms on Genz test suite, in dimension 4.

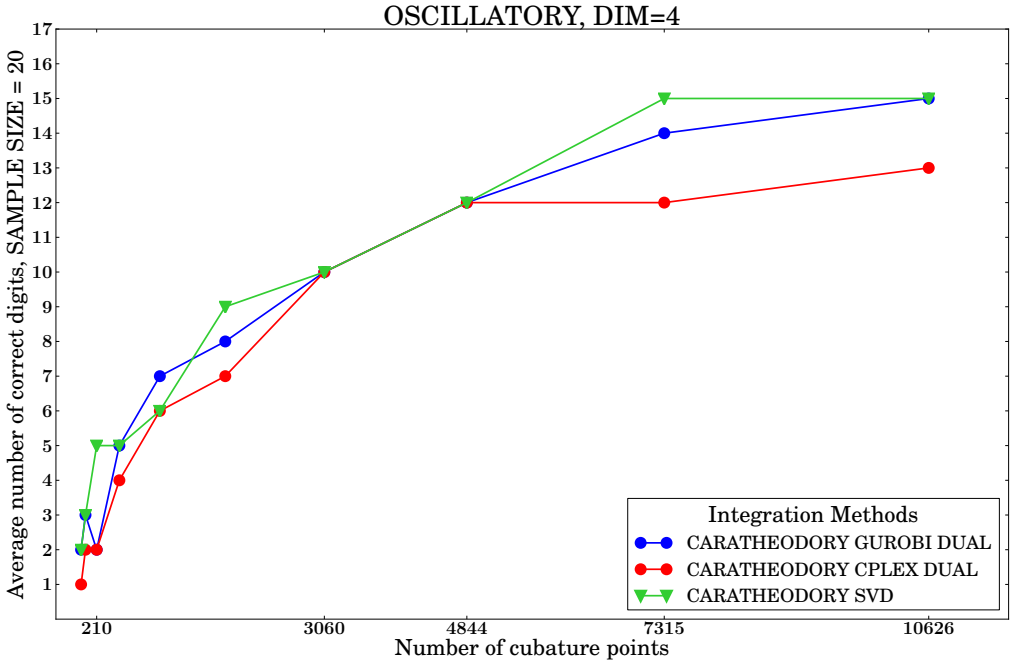


Figure 5.22: Oscillatory, Dim=4

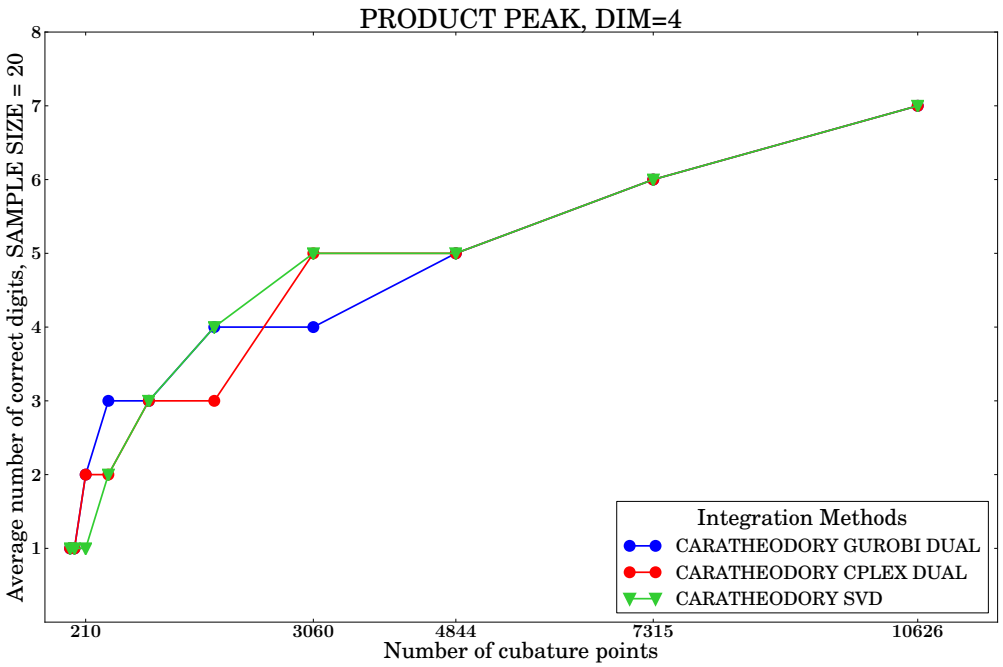


Figure 5.23: Product Peak, Dim=4

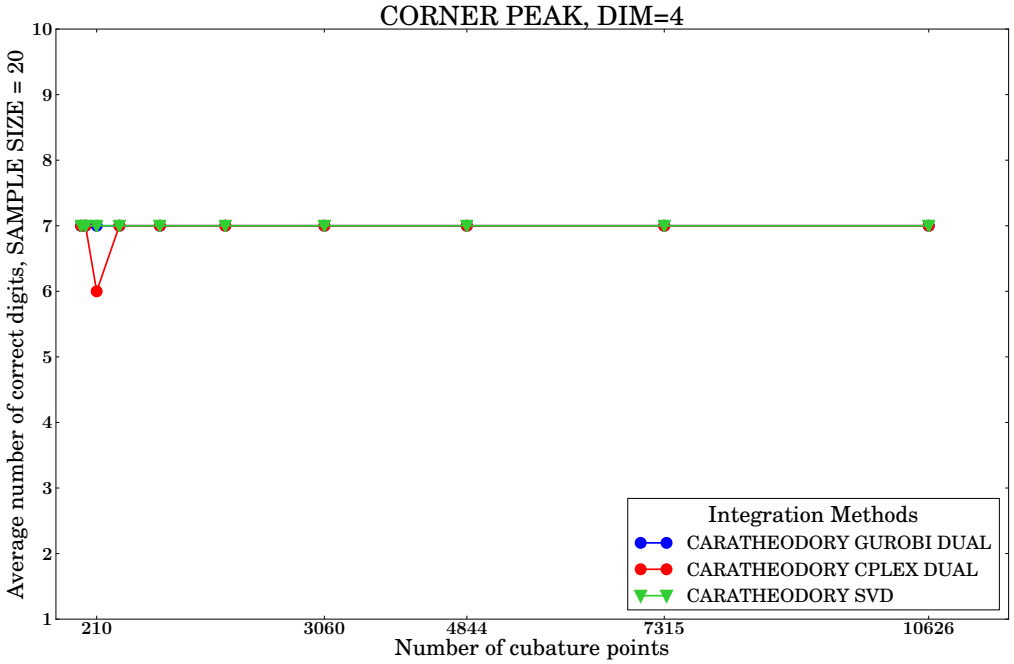


Figure 5.24: Corner Peak, Dim=4

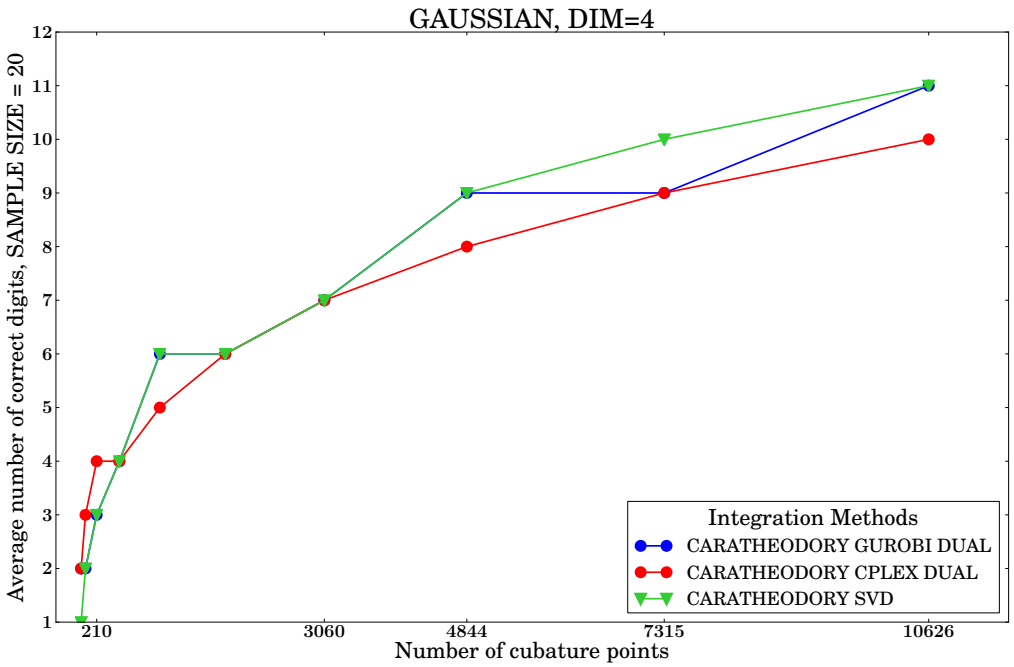


Figure 5.25: Gaussian, Dim=4

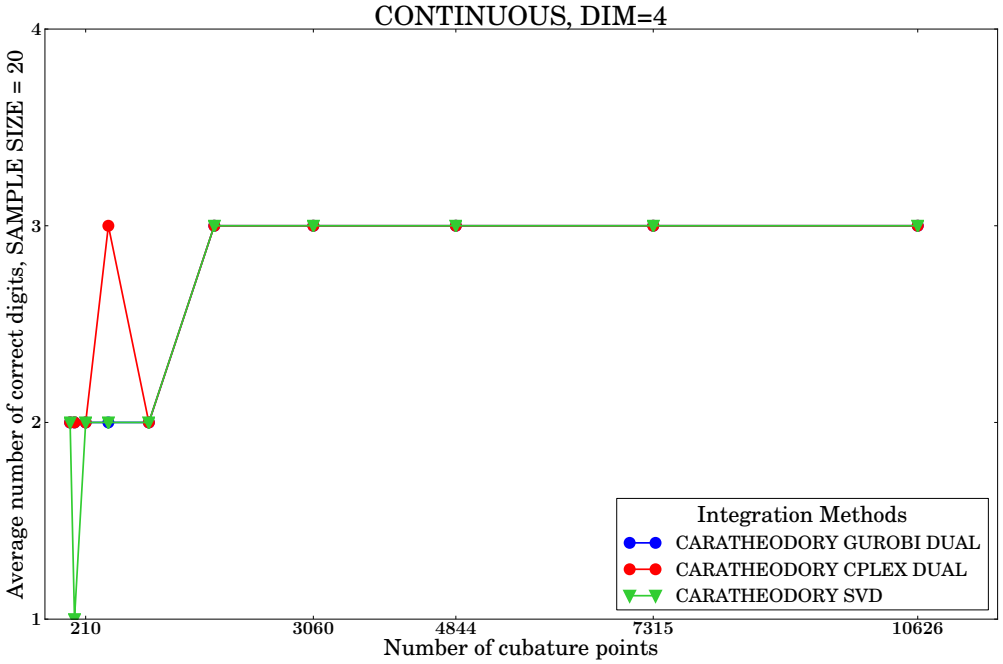


Figure 5.26: Continuous, Dim=4

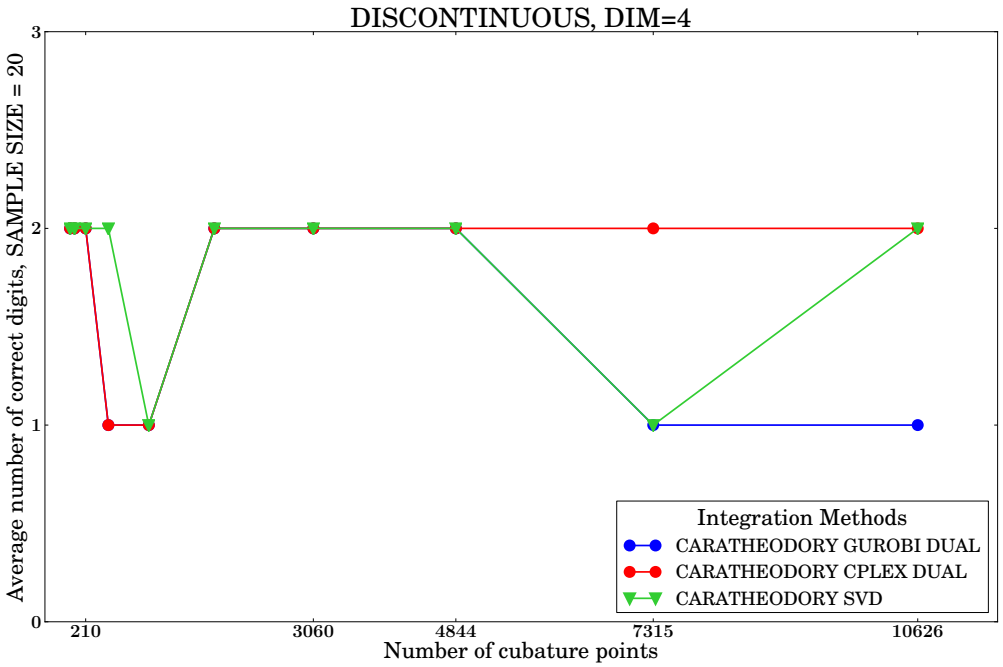


Figure 5.27: Discontinuous, Dim=4

As opposed to the Dual and Primal Simplex Algorithm implementations in Gurobi Optimizer 6.5 and IBM CPLEX Optimization Studio 12.6.2, the Interior Point Method, called the barrier method is parallelised in both packages. The IBM CPLEX Interior Point method implemented is the Barrier Optimiser, which exploits a primal-dual logarithmic barrier algorithm to generate a sequence of strictly positive primal and dual solutions.

In the following we present a comparison of Carathéodory Cubature Algorithm with again three measure reduction implementations: Gurobi Barrier LP Solver, IBM CPLEX Barrier LP Solver and Intel MKL SVD Solver. We perform this comparison in multithreaded environment with 32 threads, and as before we measure the total number of wall-clock ticks, floating point operations, resident memory and virtual memory. We provide this comparison primarily to illustrate the advantages of utilising the INTEL MKL SVD Solver in a multithreaded environment. We note that successful execution, for the data sample in question, after dimension $d = 12$ for both the Gurobi Barrier LP Solver and the IBM CPLEX LP Solver, is intermittent. By this, we mean that some cubature computations for dimensions higher than $d = 11$, do not finish execution, as a result of numerical errors, and thus cannot be relied upon to produce the required solution. See below, the typical numerical errors produced by both methods

```
Iteration    Objective       Primal Inf.    Dual Inf.      Time
    3427    0.0000000e+00   1.092270e+04   0.000000e+00   5795s
    3527    0.0000000e+00   2.358362e+03   1.911640e+13   6436s
    3627    0.0000000e+00   3.767922e+01   1.563105e+12   7158s
    3727    0.0000000e+00   3.899398e+02   3.527970e+12   7780s
    3827    0.0000000e+00   1.351013e+02   6.746612e+11   8535s

Stopped in 3904 iterations and 9143.00 seconds
Numeric error
Error code inside gurobi  = 10005
```

Figure 5.28: Numerical Errors Gurobi Barrier LP Solver

```
Summary statistics for Cholesky factor:
    Rows in Factor»      »       = 450678
    Integer space required  »    = 30596609
    Total non-zeros in factor »  = 89357347512
    Total FP ops to factor »     = 1607884747656397
CPLEX Error 1001: Out of memory
Barrier time = 103984858.45 sec
```

Figure 5.29: Numerical Errors IBM CPLEX Barrier LP Solver

We note that Gurobi Barrier LP Solver, similarly to Gurobi Dual Simplex LP Solver was stopped during execution, as the cubature computation for dimension $d = 13$ was taking longer than 200 hours to execute. Thus, we employ cross signs to indicate the lower bound on the number of ticks that would result from computations of cubature in dimensions $d = 13, 14, 15$, if each were to execute in $200, 300, 400$ hours respectively.

From these results, we note that the Gurobi Barrier LP Solver has the lowest runtime up to $d = 10$, however similarly to the Gurobi Dual Simplex LP Solver its performance degrades rapidly after the problem size of $d = 13$ is reached. In contrast, the IBM CPLEX Barrier LP Solver and the Intel MKL SVD Solver scale well. The Intel MKL SVD Solver has a higher execution runtime up to $d = 11$, however its performance remains almost constant for higher dimensions. Similar conclusions can be drawn by looking at memory consumption.

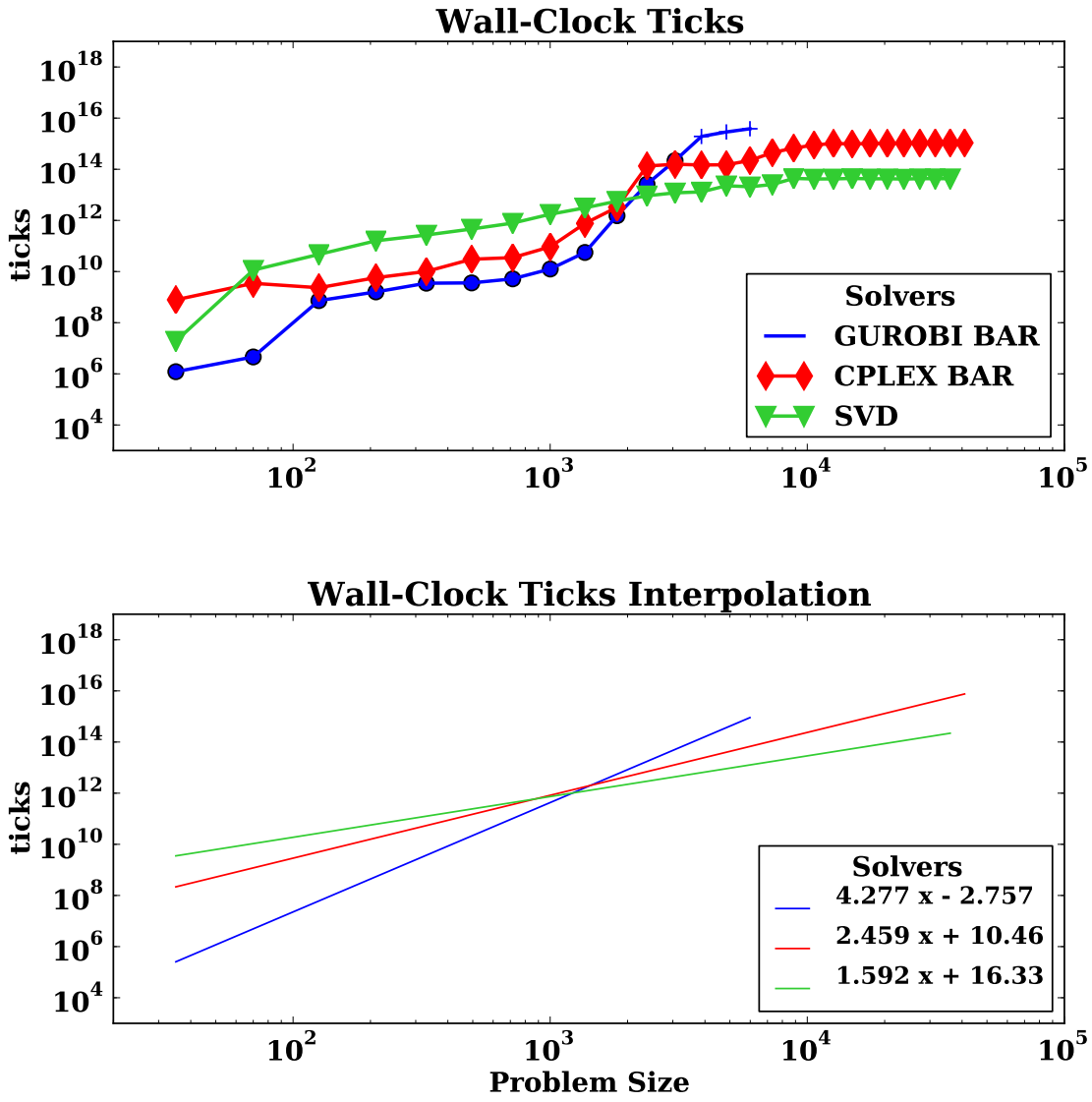


Figure 5.30: Log-log plot of Wall-Clock ticks: Gurobi Barrier, IBM CPLEX Barrier, SVD

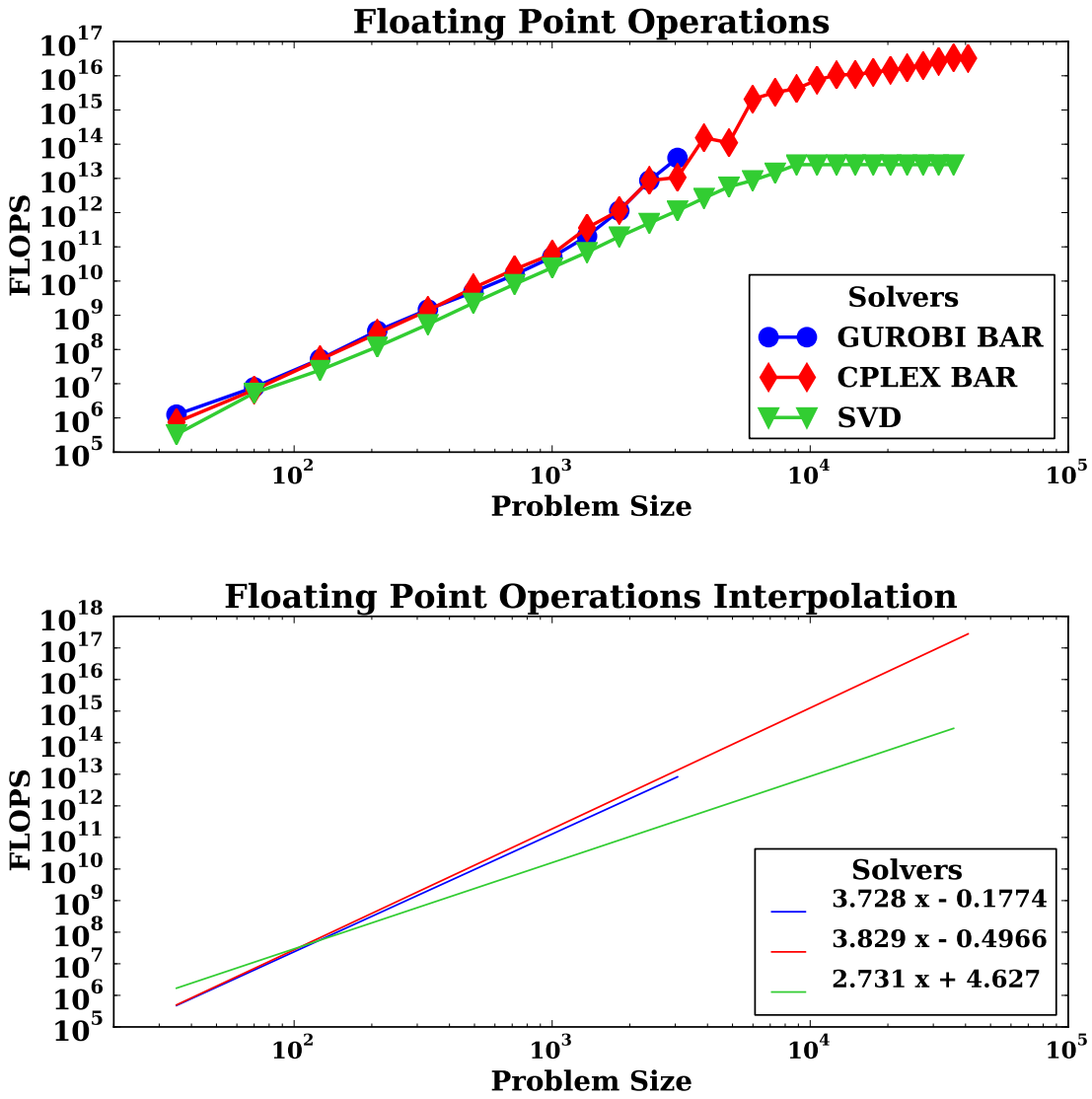


Figure 5.31: Log-log plot of Floating Point Operations: Gurobi Barrier, IBM CPLEX Barrier, SVD

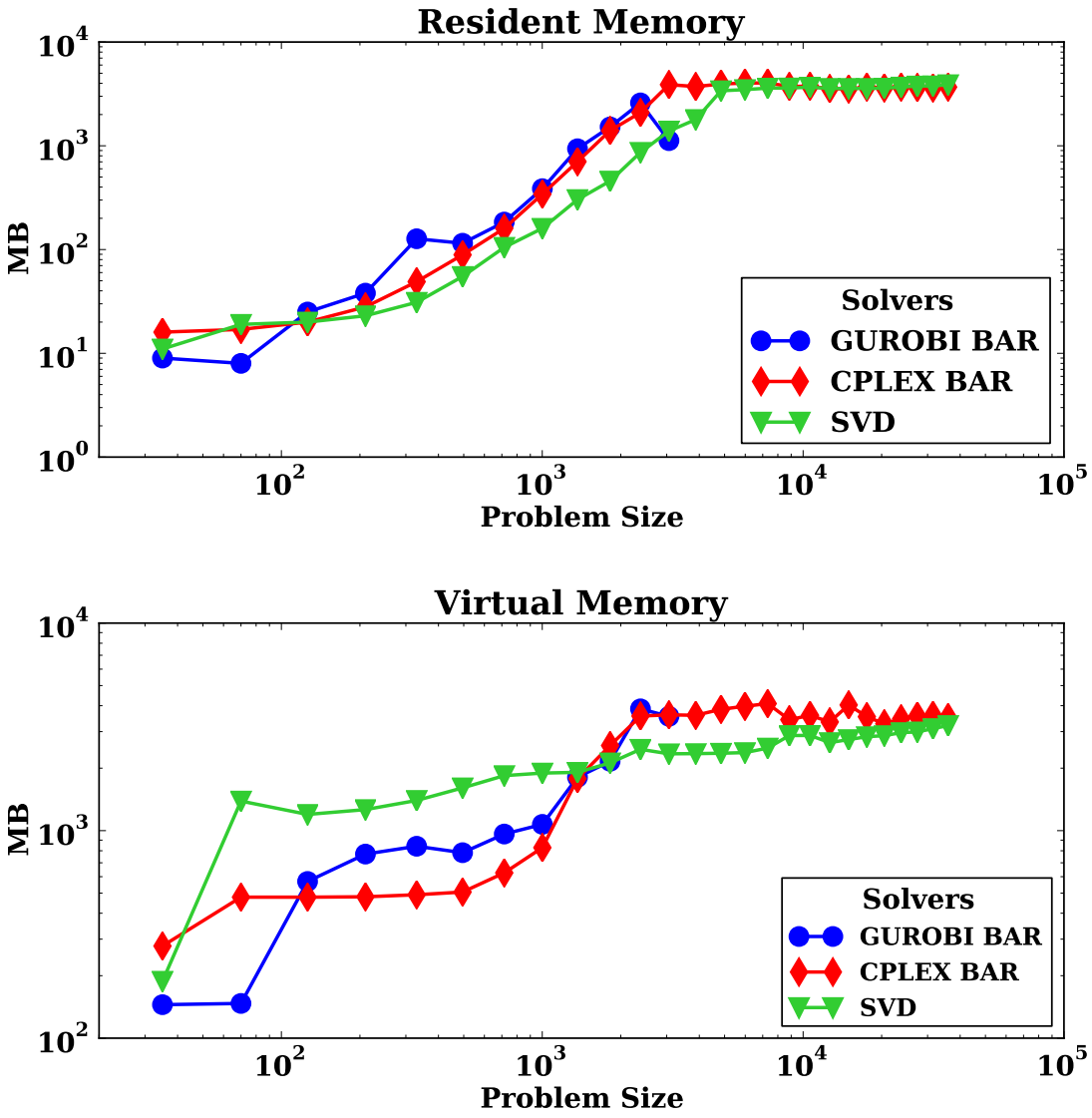


Figure 5.32: Resident and Virtual Memory: Gurobi Barrier, IBM CPLEX Barrier, SVD

5.0.8 Scalability of Carathéodory Cubature Algorithm with SVD Solver

Finally we illustrate the scalability of the Carathéodory Cubature Algorithm with the INTEL MKL SVD Solver, in terms of the runtime and floating point operations executed. Let $T(p) = T(p, M)$ denote the total wall-clock ticks of the algorithm on p processors, for some problem size M. Then, for a fixed problem size M, the speed-up of the algorithm is given by $S(p) = \frac{T(1)}{T(p)}$. By Amdahl's law or law of diminishing returns, we know that

$$S(p) \leq \frac{1}{C + \frac{1-C}{p}}$$

where C denotes the inherently sequential portion of the computation. By measuring $S(p)$ for different problem sizes M, we can hope to estimate C. In the following, we provide two example illustrations for $M = 8568$, $M = 11268$, which appear to be consistent with a broader sampling range, confirming that $C \approx 0.55$. That is to say, the estimated speed-up curves $S(p)$ are well-approximated by the Amdahl's curve

$$A(p) = \frac{1}{0.55 + \frac{0.45}{p}}$$

In particular, this sheds light on the limiting behaviour of scaling the algorithm, neglecting the overhead associated with thread creation

$$\lim_{p \to \infty} A(p) = 1.818181...$$

In addition to the scalability of the execution time, we have also measured the total number of floating point operations offloaded by the master thread to each slave thread, during the exception o f the algorithm. We observe that the percentage of the total number of flops offloaded to each slave thread is consistent between threads and consistent for different problem sizes. Indeed we observe the following scalability of flops: 60% of flops are always executed on the master thread, thereby accounting for the sequential portion of the execution. The parallel portion of the execution accounts for 40% of flops and spreads evenly between all the slave threads and the master thread. For example with two processors, the master thread executes the sequential portion of the computation and half of the parallel portion, executing approximately $60\% + 20\%$ of the workload and the remaining 20% is offloaded to the slave thread. Empirically, we observe that given k threads, $\left(60 + \frac{40}{k}\right)\%$ of flops, are executed on the master thread and $\frac{40\%}{k}$ on every slave thread. It transpires, therefore, that the scalability of the execution time is consistent with the scalability of the floating point operations, with both accounting for $55 - 60\%$ of the sequential execution.

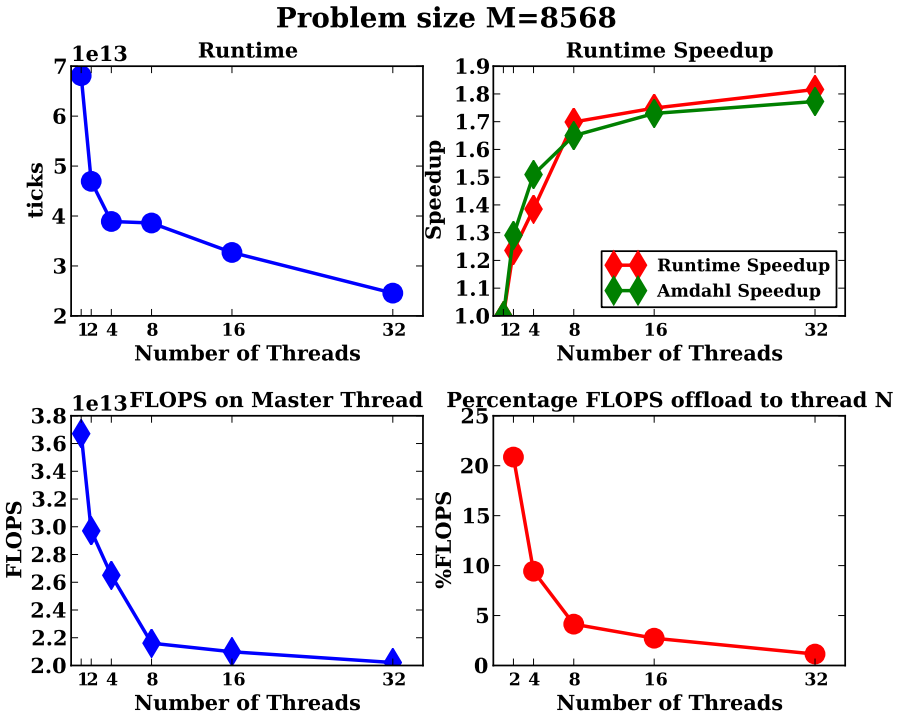


Figure 5.33: Threading Performance, Problem Size M=8568

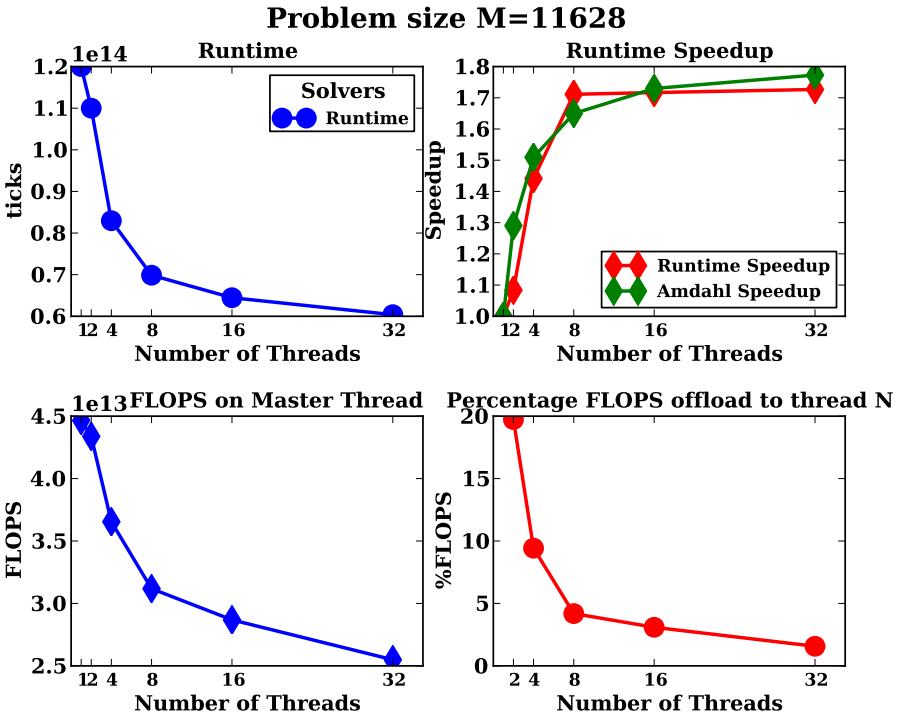


Figure 5.34: Threading Performance, Problem Size M=11628

5.1 Conclusions

We conclude that the Carathéodory Cubature Algorithm detailed in Chapter 4 is an efficient method for constructing interpolatory cubature measures in moderate dimensions. We emphasize that the Carathéodory Cubature Algorithm overcomes the "curse of dimensionality " in its construction, by thinning the support of the Cartesian Cubature product measures, whose cardinalities grow exponentially with dimension, in polynomial runtime. As described in Chapter 3, it accomplishes this, by recursively applying the Recombination Algorithm, which employs a binary tree forest data structure to store cubature nodes and weights and further employs a series of binary tree maniputations: tree splitting and updating, whilst recursively exploiting the Carathéodory Convex Hull Theorem .

The novelty of the Carathéodory Cubature Algorithm lies in the central routine of the Recombination Algorithm: M-Tree Measure Reduction, dominating the cost of the entire Carathéodory Cubature Algorithm. The latter routine, based on the Singular Value Decomposition, provides an alternative method to the Simplex and Interior Point Algorithms to compute a basic feasible solution to the underlying LP problem at each iteration through the Recombination Algorithm.

There are multiple benefits of employing the SVD-based M-Tree Measure Reduction, as demonstrated by the empirical evidence in Chapter 5. Firstly, we note that the SVD-based M-Tree Measure Reduction admits a good degree of scalability, as opposed to the inherently sequential Simplex Algorithm. Also as noted in Chapter 5, even in a single threaded environment the SVD-based M-Tree Measure Reduction outperforms the Simplex Algorithm, whose performance rapidly degrades, when a sufficiently large problem size is reached. Whilst the accuracy of the SVD-Based M-Tree Measure Reduction is comparable to that of the Simplex method, its runtime performance is at least an order of magnitude smaller and facilitates the computation of much larger problems.

In terms of efficiency, the results in Chapter 4 demonstrate that Carathéodory Cubature measures are very competitive with Smolyak and Delayed Smolyak sparse grid cubatures in moderate dimensions. Indeed, for the Genz test suite we show that Carathéodory Cubature measures generally yield twice the accuracy for the same number of cubature nodes as the ordinary Smolyak Cubature construction and a similar accuracy to the Delayed Smolyak cubature construction. We also note that the absence of negative weights in Carathéodory Cubature measures, makes them more appealing

for dealing with positive and probability measures and also in the context of iterative methods, whose stability is controlled by the norm of cubature measures.

Areas for further research and investigation should include testing the efficiency of Carathéodory Cubature measure for higher degrees and dimensions. Additionally we note that, one can easily construct sequences of nested cubature measures by the means of Carathéodory Cubature Algorithm, thus it may be interesting to investigate the idea of utilising the Smolyak sparse grid construction to produce high degree cubature formulae from moderate degree Carathéodory Cubature formulae. Moreover, an adaptive integration algorithm derived from Carathéodory Cubature construction could be explored.

References

[1] P.J. Davis: *Interpolation and Approximation.* Blaisdell, New York, (1963)

[2] C.W. Ueberhuber: *Numerical Computation.* Springer-Verlag, Berlin, Heidelberg, (1997)

[3] A.R. Krommer C.W. Ueberhuber: *Computational Integration.* SIAM, (1987)

[4] H. Engels: *Numerical Quadrature and Cubature.* Academic Press, New York, (1980)

[5] L. Fejer: *Mechanische Quadraturen mit positiven Cottesschen Zahlen.* Math.Z. 37 (1933), 287-309

[6] D. Elliott: *Truncation errors in two Chebyshev series approximations.* Math. Comp. 19 (1965), 234-248.

[7] T.J. Rivlin: *The Chebyshev Polynomials.* Wiley, New York (1974)

[8] J.P. Imhof: *On the method for numerical integration of Clenshaw and Curtis.* Numer. Math. 5 (1963) 138-141

[9] I.H. Sloan, W.E. Smith: *Properties of interpolatory product integration rules.* SIAM J. Numer.Anal. 19 (1982) 427-442

[10] L.N. Trefethen: *Is Gauss quadrature better than Clenshaw-Curtis.* SIAM Rev 50(1) (2008) 67-87

[11] A. Quateroni, R.Sacco, F.Saleri: *Numerical Mathematics.* Texts in Applied Mathematics, Springer, (2006)

[12] E. Isaacson, H.B. Keller: *Analysis of Numerical Methods.* Wiley, New York, (1966)

[13] W. Gautschi: *Orthogonal Polynomials: Applications and Computation.* Acta Numerica, (1996) 45-119

[14] I. H. Sloan, S. Joe: *Lattice methods for multiple integration.* Clarendon Press, Oxford (1994)

[15] A. Neumaier: *Introduction to Numerical Analysis.* Cambridge University Press, (2001)

[16] V. Tchakaloff: *Formules de cubature mecanique a coefficients non negatifs.* Bull. Sci. Math. 81, (1957), 123-134,

[17] P.J. Davis: *A construction of nonnegative approximate quadratures.* Math. Comp., 21 (1967), 578-582

[18] R.E. Caflisch: *Monte Carlo and quasi-Monte Carlo methods.* Acta Numerica (1998), 1-49

[19] E. Novak and K. Ritter: *High Dimensional Integration of Smooth Functions over Cubes* Numer. Math. (1996) Vol. 75 79-97

[20] A.H. Stroud: *Approximate Calculation of Multiple Integrals.* Prentice-Hall, Englewood Cliffs, NJ

[21] R. Cools and A. Haegemans: *Algorithm 824: CUBPACK: A Package for Automatic Cubature; Framework Description* ACM Trans. Math. Softw. (2003)

[22] R. Piessens, E. de Doncker, C. Ueberhuber, D. Kahaner: *Quadpack- A subroutine package for automatic integration* Springer-Verlag (1983)

[23] J.H. Halton: *On the efficiency of certain quasi-random sequences of points in evaluating multidimensional integrals* Numer. Math. 2 (1960) 84-90

[24] I.M. Sobol: *The distribution of points in a cube and the accurate evaluation of integrals* Zh. Vychisl. Mat.Mat. Fiz. 7 2 (1967) 784-802

[25] H. Niederreiter: *Random number Generation and Quasi Monte-Carlo Methods* SIAM, Philadelphia (1992)

[26] E. Novak, K. Ritter: *Simple Cubature Formulas with High Polynomial Exactness.* Constructive Approximation(1999) Vol.15, 499-522

[27] J. Berntsen, T.O. Espelid and A.Genz: *An adaptive Algorithm for the approximate calculation of multiple integrals* ACM Trans. Math. Softw.(1991)

[28] A.C. Genz and A.A. Malik: *Remarks on algorithm 006: An adaptive algorithm for numerical integration over N-dimensional rectangular region* J. Comp. Appl. Math. (1980), 295-309

[29] R. Cools and P. Rabinowitz: *Monomial Cubature Rules since " Stroud": a compilation.* J. Comput. Appl. Math. 48 (1993) 309-336

[30] R. Cools : *An Encyclopedia of Cubature Formulas* J. Complexity 19 (2003) 445-453

[31] T. Soerevik, T.O. Espelid : *Fully symmetric integration rules for the 4-cube* BIT 29 (1989), 148-153

[32] T.O. Espelid: *On the construction of good symmetric integration rules* SIAM J. Numer. Anal. 24 (1987), 855-881

[33] A. Haegemans, R. Cools: *Construction of three-dimensional cubature formulae with points on regular and semi-regular polytopes* , Numerical Integration- Recent Developments, Software and Applications (1987), 153-163

[34] A. Genz: *Fully symmetric interpolatory rules for multiple integrals* , SIAM J. Numer. Anal. 23 (1986), 1273-1283

[35] S.L. Sobolev: *Cubature formulas on the sphere invariant under finite groups rotation* , Soviet Math. 3 (1962), 1307-1310

[36] S.A. Smolyak: *Quadrature and interpolation formulas for tensor products of certain classes of functions* In Dokl. Akad. Nauk SSSR, (1963) volume 4, 240-243

[37] H.M. Moller: *Kubaturformeln mit minimaler Knotenanzahl* Numer. Math, 25 (1976), 185-200

[38] H.M. Moller: *On the construction of cubature formulae with few nodes using Groebner bases* Numerical Integration- Recent Developments, Software and Applications (1987), 177-192

[39] H.J. Schmid: *On cubature formulae with a minimal number of knots* Numer. Math. 31 (1978), 281-297

[40] S.A. Smolyak: *Quadrature and interpolation formulas for tensor products of certain classes of functions* Dokl. Akad. Nauk. SSSR, 4 (1963), 240-243

[41] T. Gerstner, M. Griebel: *Numerical Integration using Sparse Grids.* Numerical Algorithms(1998) Vol.18, 209-2322

[42] K. Petras: *Asymptotically minimal Smolyak Cubature.* Citeseer(2001)

[43] E. Novak, K. Ritter: *Simple cubature formulas with high polynomial exactness.* Constructive Approximation, 15 (1999) 499-522

[44] E. Novak, K. Ritter: *The curse of dimension and a universal method for numerical integration* Multivariate Approximation and Splines, (1998)

[45] E. Novak, K. Ritter A. Steinbauer: *A multi scale method for the evaluation of Wiener integrals* Approximation theory IX, Volume 2: Computational Aspects (1998) , 251-258

[46] G. Baszenski, F. Delvos: *Multivariate Boolean Midpoint rules* Numerical Integration IV, ISNM 112, Birkhauser (1993) , 1-11

[47] J. Burkhardt, C. Webster: *Slow Exponential Growth for Clenshaw Curtis Sparse Grids* (2012)

[48] E.A. DeVuyst, P.V. Preckel : *Gaussian Cubature: A Practitioner's Guide* Math. Comp. Model. Elsevier (2007)

[49] R.J. Vanderbei: *Linear Programming: Foundations and Extensions* Springer (2008)

[50] M.S. Bazaraa, J.J. Jarvis, H.D. Sherali: *Linear Programming and Network Flows* Wiley (2010)

[51] D.J. Rader: *Deterministic Operations Research* Wiley (2010)

[52] J.J. Modi: *Parallel Algorithms and Matrix Computation* Clarendon Press (1988)

[53] J. Dongarra, D. Lafarenza, S. Orlando: *Recent Advances in Parallel Virtual Machine and Message Passing Interface* 10^{th} European PVM/MPI User's Group Meeting Proceedings (2003)

[54] A.H. Stroud: *Approximate Calculation of Multiple Integrals.* Prentice-Hall, Englewood Cliffs, NJ

[55] M. Abramowitz and I.A. Stegun: *Handbook of Mathematical Functions: with Formulas, Graphs and Mathematical Tables* Dover, N.Y. (1970)

[56] N.S. Bahvalov: *On Approximate Calculations of Multiple Integrals.* Vestnik Moscov. Univ. Ser. Mat. Meh. Astr. Fiz. Him., 4 (1959) 3-18

[57] R. Cools and P. Rabinowitz: *Monomial Cubature Rules since " Stroud": a compilation.* J. Comput. Appl. Math. 48 (1993) 309-336

[58] R. Cools: *An Encyclopedia of Cubature Formulas* J. Complexity 19 (2003) 445-453

[59] J. Stoer and R. Bulirsh: *Introduction to Numerical Analysis* Springer-Verlag (1993)

[60] R. Cools: *Cubature Formulae and Orthogonal Polynomials* J. Comp. Appl. Math 127 (2001) 121-152

[61] J.B. Laserre: *Existence of Gaussian Cubature Formulas* J. Approx. Theory Elsevier (2012) 164(5) 572-585

[62] E. Novak and K. Ritter: *High Dimensional Integration of Smooth Functions over Cubes* Numer. Math. (1996) Vol. 75 79-97

[63] M. Chen, S. Mehrotra, D. Papp: *Scenario Generation for Stochastic Optimisation Problems via Sparse Grid Method* Numer. Math. (1996) Vol. 75 79-97

[64] M. Holtz: *Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance* Doctoral Thesis (2008)

[65] T. Gerstner and M. Griebel: *Dimension-Adaptive Tensor-Product Quadrature.* Computing (2003) Vol.71 65-87

[66] R.E. Caflish: *Monte Carlo and Quasi-Monte Carlo Methods.* Acta Numerica (1998) 1-49

[67] V.N. Temlyakov: *Cubature Formulas, discrepancy and non-linear approximation.* J. Complexity 19 (2003) 352-391

[68] N. Victoir: *Asymmetric Cubature Formulae with Few Points in High Dimension for Symmetric Measures* SIAM J. Numer. Anal. 42 (2004) 209-227

[69] A. Hinrichs and E. Novak: *Cubature Formulas for Symmetric Measures in Higher Dimensions with Few Points* J. Math. Comp. 76 (2007) 1357-1372

[70] E.A. DeVuyst and P.V. Preckel: *Gaussian Cubature: A practitioner's guide* Math. Comp. Model. 45 (2007) 787-794

[71] N. Bourbaki: *General Topology: Chapters 5-10* Springer (1998) Ettore Majorana International Science Series

[72] J. Berntsen, T.O. Espelid and A.C. Genz: *An Adaptive Algorithm for the Approximate Calculation of Multiple Integrals* ACM Trans. Math. Softw. (1991) 17-4

[73] A.C. Genz and A.A. Malik: *Remarks on Algorithm 006: An Adaptive Algorithm for Numerical Integration over an N-dimensional Rectangular Region* J. Comput. Appl. Math. 6-4 (1980) 295-309

[74] R. Cools and A. Haegemans: *Algorithm 824: Cubpack: A Package for Automatic Cubature; Framework Description* ACM Trans. Math. Softw. (2003) 29-3

[75] J. Rodriguez, D. C. Thompson, J. S. M. Anderson, J. W. Thompson, P. W. Ayers: *A Physically Motivated Sparse Cubature Scheme with Applications to Molecular Density-Functional Theory* J. Phys. A: Math Theor. (2008) 41 365202

[76] C. Berg: *Moment Problems and Orthogonal Polynomials.* http://www.math.ku.dk/kurser/2013-14/blok3/moment (2011)

[77] N.I. Akhiezer: *The Classical Moment Problem and Some Related Questions in Analysis.* Oliver and Boyd (1965)

[78] L.C. Petersen: *On the Relation between the Multidimensional and the One-dimensional Moment Problem.* Math. Scand. (1982) 361-366

[79] P.E. Gill, G.H. Golub, W.Murray, M.A. Saunders: *Methods for Modifying Matrix Factorisations* Mathematics of Computation(1974) Vol. 28, No. 126, 505-535

[80] G.W. Stewart: *An Updating Algorithm for subspace tracking* Signal Processing, IEEE Transactions(1992) Vol. 40, No. 6, 1535-1541

[81] G.W. Stewart: *Updating a Rank-Revealing ULV Decomposition* SIAM J. Matrix Anal. and Appl.(1993), Vol. 14, No. 2, 494-499

[82] E.M. Beale: *Cycling in the Dual Simplex Method* Naval Research Logistcs Quarterly(1955) Vol.2, No.4, 269-276

[83] Q.M. Tariq: *Concerning Polynomials on the Unit Interval* Proc. Amer. Math. Soc. 99 2 (1987)

[84] M. Putinar: *A Note on Tchakaloff's Theorem.* Proc. Amer. Math. Soc. (1997) Vol. 125, No. 8, 2409-2414

[85] R.T. Rockafellar: *Convex Analysis.* Princeton University Press (1996)

[86] C. Bayer and J. Teichmann: *The Proof of Tchakaloff's Theorem.* Proc. Amer. Math. Soc. (2006) Vol. 134, No. 10, 3035-3040

[87] E.M. Stein: *Singular Integrals and Differentiability Properties of Functions.* Princeton University Press (1971)

[88] W.T Shaw: *New Methods for Managing "Student's" T Distribution.*

[89] H. Brass, K. Petras: *Quadrature Theory: The Theory of Numerical Integration on the Compact Interval.* Mathematical Surveys and Monographs(2011), Vol. 178

[90] P.J. Davis, P. Rabinowitz: *Methods of Numerical Integration.* Academic Press(1975)

[91] E. Novak, H. Wozniaskowski: *When are Integration and Discrepancy tractable? .* Foundations of Computational Mathematics(2001)

[92] E. Novak, K. Ritter: *Simple Cubature Formulas with High Polynomial Exactness.* Constructive Approximation(1999) Vol.15, 499-522

[93] T. Gerstner, M. Griebel: *Numerical Integration using Sparse Grids.* Numerical Algorithms(1998) Vol.18, 209-2322

[94] E. Novak, K. Ritter: *The Curse of Dimension and a Universal Method for Numerical Integration* Multivariate Approximation and Splines(1998)

[95] S. Haber: *Numerical Evaluation of Multiple Integrals.* SIAM review(1970) Vol.12, No. 4

[96] T. Lyons and C. Litterer: *High Order Recombination and an Application to Cubature on Wiener Space.* The Annals of Applied Probability (2012) Vol.22, No. 4, 1301-1327

[97] P.E. Gill and G.H. Golub, W. Murray and M.A. Saunders: *Methods for Modifying Matrix Factorizations.* Mathematics of computation (1974) Vol. 28, No. 126, 505-535

[98] G.W. Stewart: *An updating algorithm for subspace tracking.* Signal Processing, IEEE Transactions (1992) Vol. 40, No. 6, 1535-1541

[99] G.W. Stewart: *Updating a Rank-Revealing ULV Decomposition.* SIAM. J. Matrix Anal. & Appl.,(1993) Vol. 14, No. 2, 494-499

[100] G.W. Stewart: *Rank Degeneracy.* SIAM J. Sci. and Stat. Comput., (1984) Vol. 5, No. 2, 403-413

[101] J.J. Modi: *Parallel Algorithms and Matrix Computation* Clarendon Press (1988)

[102] J.W. Demmel: *Applied Numerical Linear Algebra* SIAM (1997)

[103] C.D. Meyer: *Generalized Inversion of Modified Matrices* SIAM J. Appl. Math. 24(3) (1973) 315-323

[104] A. Bjorck: *Numerical Methods for Least Squares Problems* SIAM (1996)

[105] S.L. Campbell and C.D. Meyer: *Generalized Inverses of Linear Transformations* Dover (1991)

[106] G.H. Golub and C.F. Van Loan.: *Matrix Computations* JHU Press (2012)

[107] I.H. Sloan and S. Joe: *Lattice Methods For Multiple Integration* Oxford Science Publications (1994)

[108] P.C. Hansen: *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion* SIAM (1987)

[109] D.S. Watkins: *Fundamentals of Matrix Computations* 2nd edition, Wiley (2004)

[110] G.H. Golub, V.Klema, G.W. Stewart: *Rank degeneracy and least squares* report TR-456, Computer Science Dept. , university of Maryland, MD(1976)

[111] H.T. Rathod, B. Venkatesh: *Gauss Legendre - Gauss Jacobi Quadrature Rules over a Tetrahedral Region* Int. Journal of Math. Analysis, Vol. 5, no.4 189-198

[112] C. Hansen: *The truncated SVD as a method for regularization.* BIT Numerical Mathematics (1987) Vol. 27, No. 4, 534-553

[113] Tony F. Chan: *Rank revealing QR factorizations.* Linear Algebra and its Applications (1987) Vol. 88-89, 67 - 82

[114] S.G.L. Desmedt: *Dimension-adaptive sparse grid for industrial applications using Sobol variances.* Master of Science Thesis (2015)

[115] J. Burkhardt: *SMOLPACK: Multidimensional quadrature using sparse grids.* http://people.sc.fsu.edu/jburkardt/csrc/smolpack/smolpack.html

[116] PAPI: *http://icl.cs.utk.edu/papi/software/*