UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Efficient New Approaches for Data-Driven Global Optimization, with Applications in Computer-Aided Design

Permalink

https://escholarship.org/uc/item/4wp671cq

Author

Alimo, Shahrouz Ryan

Publication Date

2017

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Efficient New Approaches for Data-Driven Global Optimization, with Applications in Computer-Aided Design

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

in

Engineering Sciences with a Specialization in Computational Science

by

Shahrouz Ryan Alimo

Committee in charge:

Professor Thomas R. Bewley, Chair
Professor Robert R. Bitmead
Professor Sanjoy Dasgupta
Professor Juan Carlos del Alamo
Professor Philip E. Gill
Professor David A. Meyer

2017

Copyright
Shahrouz Ryan Alimo, 2017
All rights reserved.

The dissertation of Shahrouz Ryan Alimo is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2017

DEDICATION

To Shahab, Shahla, Atrshan, Shervin, Ali, Firouz.

EPIGRAPH

I know I can. Dream Big.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Prof. Thomas R. Bewley for the continuous support during my Ph.D. studies and research. His enthusiasm, motivation, and immense knowledge have helped in innumerable ways. His guidance has contributed to my personal growth, as a person and as a professional. I could not have imagined a better advisor and mentor during the development of my research. For him, I have nothing but the utmost respect and gratitude.

I would also like to thank the rest of my thesis committee: Prof. Robert Bitmead, Prof. Sanjoy Dasgupta, Prof. Juan Carlos del Alamo, Prof. Philip E. Gill, and Prof. David A. Meyer for their time and feedback. Also, I would like to thank very much my M.Sc. advisor, Prof. Alison L. Marsden, for introducing me to the derivative-free optimization and many of its applications in the Cardiovascular surgeries. Also, I would like to thank Prof. Jan P. Kleissl for helping me develop a deeper understanding about Machine Learning and its applications in Solar Energy. A special thanks goes to my mentor Dr. Firouz M. Naderi for his mentorship during these years.

I thank my Professors, mentors, and colleagues: Prof. Sergei Chernyshenko, Prof. Paolo luchini, Prof. Michael Jolly, Prof. Yousef Bahadori, Prof. Alan Hargens, Prof. Yousef Ghaffari, Prof. Mir Abbas Jalali, Prof. Amir Shamloo, Prof. Kekhosro Firouzbakhsh, Prof. Reza Mansouri, Anousheh Ansari, Prof. Dariush Divsalar, Prof. Gilbert Strang, and Prof. Behrokh Khoshnevis.

I specially thank very much my friends and lab mates: Pooriya Beyhaghi, Ramtin

Madani, Daniele Cavaligieri, Gianluca Meneghello, Muhan Zhao, Abhishek Subramanian, Elizabeth Wong, Sacha Ghebali, Bahman Moraffah, Ben Kurz, Chi Wai Chow, Bryan Urquhart, Mohamed Ghonima, Matthew Lave, Juan L. Bosch, Dipak Sahu, Oytun Babacan, Abdulelah Habib, Benjamin Gwynn, Felipe Mejia, Zack Pecenak, Ryan Hanna, Andu Nguyen, Guang Wang, Negin Nazarian, Iman Gohari, Masoud Jalali, Abhay Bangalore Ramachandra, Mahdi Esmaily Moghadam, Chris Long, Danaile Schiavazzi, Jessica Oakes, Ethan Kung, Weiguang Yang, Sethuraman Sankaran, Dibyendu Sengupta, Matthew Bockman, Ashakn Jafarpour, Mohammad Khabbazian, Payam Banazadeh, Farshid Roumi, Arash Delijani, Eduardo Ramírez, Pedro Franco, Robert Moroto, Ali Mashayek, Lukas Nonnenmacher, Ashish Cherukuri, Shuxia Tang, Adeleh Yarmohammadi, Jacqueline Papazian, Parastou Rahmani, Farzaneh Farhangmehr, Avid Boustani, Khashayar Roholahi, Sepideh Mahabadi, Mohammad Bavarian, Kiana Salari, Amir Pirbadian, Saeed Mehraban, Pantea Khodami, Amir Sadjadpour, Hootan Nikbakht, Hossein Lotfi, Hadi Pouransari, Hossein Boustanian, Sahand Pirbadian, Mahdi Alimohammadi, Fatemeh Azimlou, Shayan Modiri, Shahyar Taheri, Koosha Mirhosseini, Hamed Amini, Reza Takapoui, Soroush Kahkeshani, Hosein Mohimani, Hessam Mahdavifar, Niema Pahlevan, Houtan Yadollahi, Sina Habibollahi, Mani Bazl, Mohammadreza Aghajani, and my all of my friends.

Also, I would like to Wei-Peng Chen, Dawei He, Jorjeta Jetcheva at Fujitsu lab of America. While I was working at Fujitsu, I gained lots of experience, which helped to enhance my technical and social skills. A special thanks goes to Ferguson construction company and all of its employees particularly Nezamdoost, Shayanmehr, Nejadbakhsh

Azadani, Sedaghati, Saeedi, and Atila Mansouri.

I would like to thank my family and mentors: my parents, Shahab and Shahla, my sister, Atrshan, my brother Shervin, my cousin Ali Alima, Hassan Alimo, Alireza Alimo, Khosrow Parsa, Vahraz Zamani, Vishpar Zamani, Vishta Zamani, Azar Barani, Zari Barani, Fariba Azad Pakdaman, Luara Alimo, Bita Alima, Mehrdad Valikhani, Ashkan Alimo, Aref Karbasi, Ali Ozhand, and I would also like to express my gratitude to my extended family, who has aided me and encouraged me throughout this endeavor, for supporting me not only during these years, but also in every moment of my life. Without their endless love and constant support, none of this would have been possible.

Coauthor acknowledgments: This thesis contains parts of the following publications:

Chapter 2 contains material that has been submitted for publication and is under review. S. R. Alimo, P. Beyhaghi, T. R. Bewley, “Deluanay-based optimization algorithm via global surrogates: non convex constraints ", *Journal of Global Optimization*. The dissertation author was the primary researcher and author of this material.

Chapter 3 is currently being prepared for submission for publication. S. R. Alimo, D. Cavaglieri, P. Beyhaghi, T.R. Bewley, “Discovery of an IMEXRK time integration scheme via Delaunay-based derivative-free global optimization ", *Journal of Global Optimization*. The dissertation author was the primary researcher and author of this material.

Chapter 4 contains material that has been submitted for publication and is under review. S. R. Alimo, P. Beyhaghi, T. R. Bewley, “Deluanay-based optimization algorithm via global surrogates incorporating derivative information ", submitted to *Control Decision Conference 2017*.

Chapter 5 is currently being prepared for submission for publication. S. R. Alimo, P. Beyhaghi, T. R. Bewley, “Implementation of dense lattices to accelerate Delaunay-based optimization.", in preparation to submit for *Optimization Methods and Software Journal*. The dissertation author was the primary researcher and author of this material.

Chapter 6 contains material that previously in part is published in: S. Alimohammadi, P. Beyhaghi, G. Meneghello, T. R. Bewley, (2017) Delaunay-based optimization

in CFD leveraging multivariate adaptive polyharmonic splines (MAPS). *Proceedings of the 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, p. 129. The dissertation author was the primary researcher and author of this material.

Chapter 7 is currently being prepared for submission for publication. P. Beyhaghi, S. R. Alimo, T. R. Bewley, "A Delaunay-based method for optimizing infinite time averages of numerical discretizations of ergodic systems". The dissertation author was the primary researcher and author of this material.

VITA

2012	B. Sc. in Mechanical Engineering *Summa cum laude*, Sharif University of Technology, Iran
2014	M. Sc. in Engineering Sciences (Mechanical Engineering), University of California, San Diego
2017	Ph. D. in Engineering Sciences with a Specialization in Computational Science, University of California, San Diego

PUBLICATIONS

Alimo, S. R., Beyhaghi, P., and Bewley. T. R. : *Delaunay-based Derivative-free Optimization via Global Surrogates, Part III: nonconvex constraints.* Journal of Global Optimization. Under review.

Alimo. S. R., Cavaglieri, D., Beyhaghi, P., and Bewley, T. R. : *Design of IMEXRK time integration schemes via Delaunay-based derivative-free optimization with nonconvex constraints and grid-based acceleration.*, *under preparation for Journal of Global Optimization*.

Alimo, S. R., Beyhaghi, P., and Bewley, T. R. : *Optimization combining derivative-free global exploration with derivative-based local refinement*, submitted to Control Decision Conference (CDC-2017), under review.

Alimo, S. R., Beyhaghi, P., and Bewley, T. R. : *Implementation of dense lattices to accelerate Delaunay-based optimization.* Under preparation for Optimization Methods and Software Journal.

Alimo, S., Beyhaghi, P., Meneghello, G., and Bewley, T. R. (2017) Delaunay-based optimization in CFD leveraging multivariate adaptive polyharmonic splines (MAPS). *Proceedings of the 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, p. 129.

Beyhaghi, P., Alimo, S. R., and Bewley, T. R. *A Delaunay-based method for optimizing infinite time averages of numerical discretizations of ergodic systems*, submitted to Control Decision Conference (CDC-2017).

Beyhaghi, P., Alimo, S. R., and Bewley, T. R. *A multiscale, asymptotically unbiased approach to uncertainty quantification in the numerical approximation of infinite time-averaged statistics.* Under preparation.

Alimohammadi, S., He, D. (2016). *Multi-Stage Algorithm for Uncertainty Analysis of Solar Power Forecasting*. IEEE Power & Energy Society General Meeting.

ABSTRACT OF THE DISSERTATION

Efficient New Approaches for Data-Driven Global Optimization, with Applications in Computer-Aided Design

by

Shahrouz Ryan Alimo

Doctor of Philosophy in

Engineering Sciences with a Specialization in Computational Science

University of California, San Diego, 2017

Professor Thomas R. Bewley, Chair

Alongside derivative-based methods, which scale better to higher-dimensional problems, derivative-free methods play an essential role in the optimization of many practical engineering systems, especially those in which function evaluations are determined by statistical averaging, and those for which the function of interest is nonconvex in the adjustable parameters.

This work focuses on the development of a new family of derivative-free, or "data-driven", optimization schemes, namely Δ-DOGS schemes. The idea unifying this efficient and (under the appropriate assumptions) provably-globally-convergent family of schemes is the minimization of a search function which linearly combines a computationally inexpensive "surrogate" (that is, an interpolation, or in some cases a regression, of recent function evaluations - we generally favor some variant of polyharmonic splines for this purpose), to summarize the trends evident in the data available thus far, with a synthetic piecewise-quadratic "uncertainty function" (built on the framework of a Delaunay triangulation of existing datapoints), to characterize the reliability of the surrogate by quantifying the distance of any given point in parameter space to the nearest function evaluations. This thesis introduces a handful of new schemes in the Δ-DOGS family: (a) Δ-DOGS(Ω) designs for nonconvex (even, disconnected) feasible domains defined by computable constraint functions within a bound search domain. (b) Δ-DOGS(Ω_Z) accelerates the convergence of Δ-DOGS(Ω) by restricting function evaluation coordinations at each iteration to lie on a Cartesian grid in a bounded search domain. The grid is successively refined as convergence is approached. (c) gradient-based acceleration of Δ-DOGS combines derivative-free global exploration with derivative-based local refinement. (d) Δ-DOGS(Λ) modifies the idea of restricting function evaluations at each iteration to lie on a dense lattice (derived from an n-dimensional sphere packing) instead of a Cartesian grid. Moreover, it handles the linear constraint domain. (e) α-DOGSX designs to simultaneously increase the sampling time, and refine the numerical approximation, as convergence is approached.

This work also introduces a method to scale the parameter domain under consideration based on adaptive variation of the seen data in the optimization process, thereby obtaining a significantly smoother surrogate. This method is called the Multivariate Adaptive Polyharmonic Splines (MAPS) surrogate model.

Practical applications of these algorithms are explored that include (A) the design of low-storage implicit/explicit Runge-Kutta (IMEXRK) schemes for high performance computing (HPC) problems such as the numerical simulation of turbulence flows, and (B) the design of airfoils and hydrofoils.

These algorithms have been compared with existing state-of-the-art algorithms, particularly the Surrogate Management Framework (SMF) using the Kriging model and Mesh Adaptive Direct Search (MADS), on both standard synthetic and computer-aided shape designs. We showed that in most cases, the new Δ-DOGS algorithms outperform the existing ones.

Chapter 1

Introduction

Failure is an option here. If things are not failing, you are not innovating enough.—Elon Musk

1.1 Overview

Delaunay-based Derivative-free Optimization via Global Surrogates (Δ-DOGS) is a generalizable family of practical, efficient, and provably-convergent derivative-free algorithms designed for a range of nonconvex optimization problems with expensive function evaluations. Algorithms in this family are a subset of response surface methods (RSMs) that iteratively minimize metrics based on a surrogate model of existing datapoints as well as a synthetic model of the uncertainty of this surrogate.

We developed a handful of schemes in this family, including schemes designed specifically for (a) simple bound constraints, (b) linear constraints, and (c) nonconvex

constraints, including those that define disconnected feasible domains. Accelerated variants of these Δ-DOGS algorithms have been proposed that, at each iteration, restrict the function evaluations to lie on Cartesian grids or dense lattices that are successively refined as convergence is approached. The interpolation strategy used in these surrogate-based optimization techniques have also been examined closely; polyharmonic splines interpolation provides a natural starting point, but improved interpolation strategies, which rescale the parameter space based on the existing function evaluations at each iteration, have also been explored. This rescaling approach facilitates, in a certain setting, the reduction of the dimension of the optimization problem considered at many iterations, thereby further accelerating convergence.

In many practical problems, the calculation of the true objective function, for any feasible set of values of the parameters in the problem, is not exact. In some cases, the true objective function that we desire to minimize is given by the infinite-time average of a statistically-stationary ergodic process; in such problems, any numerical or experimental approximations of this function is characterized by sampling error, which may be reduced by additional sampling. For problems of this type, a variant was designed to simultaneously increase the statistical sampling and refine the numerical approximation of the function evaluations as convergence is approached, dubbed α-DOGSX [1]. This unique algorithm is specifically designed to efficiently minimize the true objective function, while using minimal sampling over the parameter space. α-DOGSX is improved for the cases where a lower estimation for the objective function is available to accel-

[1] α-DOGSX has been adapted from α-DOGS algorithm [3].

erate the convergence results. The key idea which α-DOGSX efficiently implements is that only a limited number of sampling is needed far from the global minimum in order to effectively "rule out" that region of parameter space, whereas more extensive and accurate sampling is required as convergence is approached in order to more precisely quantify the objective function in regions closer to the global minimum.

Chapter 2 introduces a Delaunay-based derivative-free optimization algorithm, dubbed Δ-DOGS(Ω), for problems with both (a) a nonconvex, computationally expensive objective function $f(x)$, and (b) nonlinear, computationally expensive constraint functions $c_\ell(x)$ which, taken together, define a nonconvex, possibly even not connected feasible domain Ω, which is assumed to lie within a known rectangular search domain L_s, everywhere within which $f(x)$ and the $c_\ell(x)$ may be evaluated. Approximations of both the objective function $f(x)$ as well as the feasible domain Ω are developed and refined as the iterations proceed. The work is an extension of our original Delaunay-based optimization algorithm (see JOGO DOI: 10.1007/s10898-015-0384-2), and inherits many of the constructions and strengths of that algorithm, including: (1) a surrogate function $p(x)$ interpolating all existing function evaluations and summarizing their trends, (2) a synthetic, piecewise-quadratic uncertainty function $e(x)$ built on the framework of a Delaunay triangulation amongst existing datapoints, (3) a tunable balance between global exploration (large K) and local refinement (small K), (4) provable global convergence for sufficiently large K, under the assumption that the objective and constraint functions are twice differentiable with bounded Hessians, and (5) remarkably fast global convergence on a variety of benchmark problems.

Chapter 3 introduces Δ-DOGS(Ω_Z) that is a modification to Δ-DOGS(Ω) to accelerate the convergence speed to a global minimizer in the practical application problems. Δ-DOGS(Ω_Z) is applied to the problem of identifying a new, low-storage, high-accuracy, Implicit/Explicit Runge-Kutta (IMEXRK) time integration scheme for high performance computing (HPC) applications, like the simulation of turbulence. The optimization scheme developed and used in this work, which is provably globally convergent under the appropriate assumptions, combines the essential ideas of (a) Δ-DOGS(Ω) algorithm, which is designed to efficiently optimize a nonconvex objective function $f(x)$ within a nonconvex feasible domain Ω described by a number of constraint functions $c_\ell(x)$, with (b) acceleration of Δ-DOGS using Cartesian grid, which aims to reduce the number of function evaluations on the boundary of the feasible domain that would otherwise be called for via the restriction that all function evaluations lie on a Cartesian grid, which is subsequently refined as the iterations proceed, over the rectangular search domain L_s considered. The identification of the optimal parameters of IMEXRK schemes involves (1) a complicated set of nonlinear constraints, which are imposed in order to achieve the desired order of accuracy in addition to a handful of important stability properties, which leads to a highly nonconvex, disconnected feasible domain, and (2) a highly nonconvex objective function, which represents a compromise between a few different measures characterizing the leading-order error and potential stability shortcomings of the resulting scheme. This structure makes the computation of new IMEXRK schemes a challenging and well-suited practical test problem for global optimization algorithms to solve. In this work, the new optimization algorithm devel-

oped, Δ-DOGS(Ω_Z), introduces the notion of "support points", which are points defined and used to eliminate constraint and objective function evaluations on the boundary of the search domain, where these functions are sometimes ill-behaved, while restricting all datapoints to like on a Cartesian grid that is successively refined as convergence is approached. For validation, the convergence of Δ-DOGS(Ω_Z) and Δ-DOGS(Ω) are compared on a challenging problem of optimizing a low-storage IMEXRK formulation. Results indicate a notably accelerated convergence rate using Δ-DOGS(Ω_Z). In the end, a low-storage third-order accurate IMEXRK algorithm for the time integration of stiff ODEs was identified which exhibited remarkably good stability and accuracy properties as compared with existing IMEXRK schemes.

Chapter 4 proposes a hybrid optimization scheme combining an efficient (and, under the appropriate assumptions, provably globally convergent) derivative-free optimization algorithm, Δ-DOGS, to globally explore expensive nonconvex functions, with a new derivative-based local optimization algorithm, to maximally accelerate local convergence from promising feasible points discovered in parameter space. The resulting hybrid optimization scheme proceeds iteratively, automatically shifting between (derivative-free) global exploration and (derivative-based) local refinement as appropriate. The new local derivative-based trust region method implemented uses the Voronoi partitioning of the existing datapoints for the construction of a trust region around the current best point. The resulting algorithm is analyzed, and its global convergence is proven under certain assumptions on the objective function. Finally, the algorithm is applied to nonconvex optimization problems with multiple local minima, and its com-

putational cost compared with that of the original Δ-DOGS algorithm.

Chapter 5 introduces an accelerated variant of our Delaunay-based derivative-free optimization strategy for feasible domains bounded by linear constraints. One of the challenges of the basic algorithms in this class is their over-exploration of the boundaries of feasibility, which slows convergence. For feasible domains with simple bound constraints, we recently introduced a method to coordinate such a Delaunay-based search by quantizing each new point to be evaluated onto a Cartesian grid which is successively refined as convergence is approach; this approach substantially accelerated convergence. For feasible domains with linear constraints, this chapter introduces a new algorithm, dubbed Δ-DOGS(Λ), to coordinate a Delaunay-based search with a "fitted lattice", which implements an n-dimensional dense lattice over the interior of the feasible domain, and successively lower-dimensional dense lattices over the boundaries of the feasible domain with increasing numbers of active constraints. Global convergence of the algorithm is proved under the appropriate assumptions, and the algorithm is validated on a number of representative test problems.

Chapter 6 presents a new interpolation strategy, dubbed multivariate adaptive polyharmonic splines (MAPS), which is proposed to mitigate this irregular behavior, thereby accelerating the convergence of Δ-DOGS. The MAPS approach modifies the polyharmonic spline (PS) approach by rescaling the parameters according to their significance in the optimization problem based on the data available at each iteration. This regularization of the PS approach ultimately reduces the number of function evaluations required by Δ-DOGS to achieve a specified level of convergence in optimization prob-

lems characterized by parameters of varying degrees of significance. The importance of this rescaling of the parameters during the interpolation step is problem specific. To quantify its beneficial impact on a practical problem, we compare Δ-DOGS with MAPS to Δ-DOGS with polyharmonic splines on an application related to hydrofoil shape optimization in seven parameters; results indicate a notable acceleration of convergence leveraging the MAPS approach.

Chapter 7 extends Δ-DOGS to efficiently solve those problems whose function evaluations are not accurate, but these inaccuracies can be improved. In many practical problems, the calculation of the true objective function, for any feasible set of values of the parameters in the problem, is not exact. α-DOGSX efficiently solves these types of problems by having a limited amount of sampling far from the minimum of the objective function and effectively "rule out" that region of parameter space, and considers more extensive sampling in regions closer to the global minimum as convergence is approached. α-DOGSX could address uncertainties of the objective function that are generated by the numerical discretization of an original ODE or PDE problem of interest. For validation, this modified optimization algorithm is applied to the (chaotic) Lorenz system. Numerical results indicate that, following the new approach, most of the computational effort is spent close to the optimal solution as convergence is approached.

1.2 Organization of the thesis

The content of this thesis can be summarized as follows:

Chapter 2 presents Δ-DOGS(Ω) as a global optimization algorithm which can minimize any nonconvex objective function inside nonconvex (and even disconnected) constrained domains.

Chapter 3 presents Δ-DOGS(Ω_Z), a modification of Δ-DOGS(Ω) that accelerates its convergence results and applies it to design new IMEXRK schemes, IMEXRKiCBA, for high performance computing.

Chapter 4 combines Δ-DOGS with local derivative-based methods for faster convergence results when the gradient information is available.

Chapter 5 presents Δ-DOGS(Λ), which solves one of the issues associated with the bound search domain and extends these algorithms to use any dense lattice within linear constrained domains.

Chapter 6 applies the Δ-DOGS(Λ) framework for efficient design of hydrofoils. Also, a new interpolation strategy called Multivariate Adaptive Polyharmonic Spline (MAPS) is introduced to deal with the practical problems which the variation of objective function non-uniformly changes respect to the design parameters over the feasible domain.

Chapter 7 develops an optimization algorithm α-DOGSX, which can minimize objective functions that are obtained by taking the infinite time-averaged statistics of a stationary ergodic process and whose error is due to finite sampling and discretization error.

Chapter 2

Delaunay-based derivative-free optimization via global surrogates with nonconvex feasible domain: Δ-DOGS(Ω)

We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard. —John F. Kennedy

2.1 Introduction

The problem considered in this paper is as follows:

$$\text{minimize } f(x) \quad \text{with} \quad x \in \Omega := L_c \cap L_s \subseteq \mathbb{R}^n \quad \text{where}$$

$$L_c = \{x | c_\ell(x) \leq 0\,, \text{ for } \ell = 1, \cdots, m\}, \quad L_s = \{x | a \leq x \leq b\}, \tag{2.1}$$

where both $f(x)$ and $c_\ell(x)$ for $\ell = 1, \cdots, m$ are twice differentiable and possibly nonconvex functions which map $\mathbb{R}^n \to \mathbb{R}$ within the search domain L_s. The optimization problem (2.1) has two sets of constraints:

a. a set of $2\,n$ bound constraints that characterize the n-dimensional box domain $L_s = \{x | a \leq x \leq b\}$, dubbed the *search domain*, and

b. a set of m possibly nonlinear inequality constraints $c_\ell(x) \leq 0$ that together characterize the possibly nonconvex domain L_c, dubbed the *constraint domain*.

The *feasible domain* is the intersection of these two domains, $\Omega := L_s \cap L_c$.

Potential applications of an efficient optimization algorithm of this type include: (a) minimizing the ratio of lift to drag, while holding the lift coefficient constant, in the design optimization of airfoils [5]; (b) minimizing entropy generation, with a constant wall temperature or wall heat flux, in the optimization of finned-tube heat exchangers [6]; (c) optimization of cardiovascular stints [7, 8]; and (d) maximizing solar power plant efficiency [9, 10].

Constrained optimization problems of the form given in (2.1), dubbed Nonlinear

Inequality Problems (NIPs), have been studied widely using both derivative-based and derivative-free optimization strategies.

In the derivative-based setting, there are two main classes of approaches for NIPs. The first, Sequential Quadratic Programming (SQP) methods (see, e.g., [11, 12]), impose a local quadratic model for the objective function and a local linear model for the constraint functions to estimate the location of the local minimum at each step. These models are defined based on the local gradient and Hessian of the objective function, and the local Jacobian of the constraint functions; thus, these methods are applicable only when either these derivatives, or accurate approximations thereof, are available. SQP methods only convergence to a KKT point, or to a local minimum, and special care is needed to guarantee such convergence. Application of SQP methods to problems with nonconvex feasible domains leads to some especially difficult technical challenges. Effective implementations of SQP methods include SNOPT [12] and SQP Filter [13].

The other main class of derivative-based approaches for NIPs is penalty methods (see, e.g., [14]), which modify the objective function by adding a penalty term which is successively refined as the optimization proceeds. In this way, a series of *unconstrained* problems is considered, these problems are iteratively adjusted as convergence is approached such that the iterative process eventually converges to a local minimizer of the original *constrained* problem. There are two main types of penalty functions used in such approaches, quadratic penalty functions (see, e.g., [15, 16, 17]) and barrier functions (see, e.g., [18]).

Methods based on quadratic penalty functions add a smooth penalty *outside* the

feasibility boundary. This penalty goes to zero at the feasibility boundary, and is made successively steeper near the feasibility boundary (that is, it increases without bound on the exterior of the feasible domain) as convergence is approached. Methods based on quadratic penalty functions do not require the identification of an initial feasible point during initialization, though the objective function is evaluated over a search domain which extends outside the feasible domain, so these infeasible objective function computations need to be well behaved.

Methods based on barrier functions add a smooth penalty *inside* the feasibility boundary. This penalty goes to infinity at the feasibility boundary, and is made successively steeper near the feasibility boundary (that is, it is diminished towards zero on the interior of the feasible domain) as convergence is approached. Methods based on barrier functions require the identification of an initial feasible point during initialization, though all subsequent function evaluations are feasible, so the objective function need not be well behaved outside the feasible domain.

Derivative-based methods, though scaling to higher-dimensional optimization problems far better than derivative-free methods, have certain distinct challenges. First, they are usually not designed to find the global minimum of a nonconvex objective function. Further, an expression (or, a numerical approximation) of the gradients of the objective and constraint functions are needed; such derivative information is often difficult or impossible to obtain.

A large number of derivative-free strategies for constrained optimization have been proposed. Some of these methods, like the downhill simplex and direct search

methods [including the generalized pattern search (GPS) and mesh-adaptive direct search (MADS) algorithms], only assure local convergence. Others, like simulated annealing, genetic algorithms, exhaustive search strategies, and response surface methods, provide global convergence; however, most such methods perform global exploration steps in either a random or an exhaustive fashion, and are thus quite inefficient with function evaluations. Response surface methods are, today, the most efficient class of optimization schemes which provide global convergence [19].

Direct search methods do not use any model for the objective or constraint functions, and the solution of the optimization problem is found based only on a series of exploratory function evaluations inside the feasible domain. GPS methods [20] are well-known algorithms in this class, which restrict all function evaluations to lie on underlying grid that is successively refined. GPS methods were initially designed for unconstrained problems, but have been modified to address bound constrained problems [21], linearly-constrained problems [22], and smooth nonlinearly constrained problems [23]. MADS methods [24, 25, 26, 27, 28] are modified GPS methods which can handle non-smooth constraints.

Response Surface Methods (RSMs) [29, 28, 30, 31, 32, 33], on the other hand, leverage an underlying inexpensive model, or "surrogate", of the objective function[1]. Kriging interpolation [34, 35, 36] was initially used to develop this surrogate. This convenient choice provides both an interpolant and a model of the uncertainty of this

[1]This approach generalizes the SQP method, where a quadratic function is used to locally model the objective function, and linear function is used to locally model the constraints.

interpolant. However, such correlation-based interpolation strategies have a number of numerical shortcomings (see, e.g., the appendix of [37]). Thus, our group has explored [1, 37] a new class of RSMs which can employ *any* interpolation strategy for the surrogate. Up to now, these approaches have been restricted to problems for which the feasible domain is convex and the constraint functions are numerically inexpensive. The present work modifies the algorithms developed in [1, 37] to address problems for which the constraint functions are nonlinear and expensive to compute and the feasible domain is nonconvex.

The structure of this paper is as follows. Section 2.2 describes the main elements of the optimization algorithm itself. Section 2.3 analyzes its convergence properties, and describes the technical conditions needed to guarantee its convergence to a global minimizer. Section 2.4 presents a procedure to adjust the tuning parameter of the algorithm developed in §2.2 to maximize the speed of convergence if an estimate of the global minimum (but, not the global minimizer) is available. Section 2.5 discusses briefly the behavior of the algorithm proposed when the feasible domain is empty. In Section 2.6, the algorithm is applied to some benchmark optimization problems to illustrate its behavior. Conclusions are presented in Section 2.7.

2.2 Algorithm

In this section, we present an algorithm to solve the optimization problem defined in (2.1), in a manner which efficiently handles inequality constraint functions $c_\ell(x)$

that are nonlinear and computationally expensive, and which (together with L_s) define a feasible domain Ω that may be nonconvex and, even, not connected.

2.2.1 Preliminary definitions

We first present some preliminary notions.

Definition 1. *Let S be a set of points in L_s, including its vertices. Define Δ as a Delaunay triangulation (see [38]) of S. Then, for each simplex $\Delta_i \in \Delta$, the local uncertainty function $e_i(x)$ is defined as*

$$e_i(x) = (r_i)^2 - \|x - z_i\|^2, \tag{2.2}$$

where z_i and r_i are the circumcenter and circumradius of the simplex Δ_i, respectively. The global uncertainty function $e(x)$ is then defined as

$$e(x) = e_i(x), \quad \forall x \in \Delta_i. \tag{2.3}$$

The functions $e_i(x)$ and $e(x)$ have a number of properties which are shown in §3 of [37], the most important of which, for the present purposes, are as follows.

Property 1. *The global uncertainty function $e(x)$ is piecewise quadratic, continuous, and Lipschitz, with Lipschitz constant $2\,R_{\max}$, where $R_{\max}$ is the maximum circumradius of the corresponding Delaunay triangulation.*

Property 2. *The Hessian of local uncertainty function $e_i(x)$ is equal to $-2\,I$.*

Property 3. *The global uncertainty function $e(x)$ is equal to the maximum of the local uncertainty functions $e_i(x)$, that is,*

$$e(x) = \max_{i=1,\cdots,E} e_i(x), \quad \forall x \in \Delta, \tag{2.4}$$

where E is the number of simplices in the triangulations.

Definition 2. *Let S be a set of (both feasible and infeasible) points in L_s, including its vertices, at which the objective function $f(x)$ has been evaluated. Consider $p(x)$ as an interpolating function in Δ that interpolates the objective function $f(x)$ at all points in S. Then, for each simplex $\Delta_i \in \Delta$, the local search function $s_i(x)$ is defined as*

$$s_i(x) = p(x) - K\, e_i(x), \tag{2.5}$$

where K is a tuning parameter. The global search function $s(x)$ is defined as

$$s(x) = s_i(x) \qquad \forall x \in \Delta_i, \tag{2.6}$$

where Δ_i is the i'th simplex in Δ. Note that

$$s(x) = \min_{i=1,\cdots,E} s_i(x) \qquad \forall x \in \Delta.$$

Definition 3. *Consider $g_1(x)$, $g_2(x)$, $\cdots$, $g_m(x)$ as interpolating functions in Δ that interpolate the constraint functions $c_1(x)$, $c_2(x)$, $\cdots$, $c_m(x)$, respectively, through the points*

in S. Then, for each simplex $\Delta_i \in \Delta$, the ℓ'th local constraint search function $\tilde{s}_{\ell,i}(x)$ is defined as

$$\tilde{s}_{\ell,i}(x) = g_\ell(x) - K\, e_i(x) \leq 0, \qquad \ell = 1, 2, \cdots, m, \tag{2.7}$$

where K is a constant tuning parameter. The ℓ'th global constraint search function $\tilde{s}_\ell(x)$ is defined as

$$\tilde{s}_\ell(x) = \tilde{s}_{\ell,i}(x), \qquad \forall x \in \Delta_i, \quad \ell = 1, 2, \cdots, m. \tag{2.8}$$

Note that, by Property 3 above,

$$\tilde{s}_\ell(x) = \min_{i=1,\cdots,E} \tilde{s}_{\ell,i}(x), \quad \forall x \in \Delta.$$

Remark 1. *The (single) constant K is a tuning parameter that specifies the trade-off between global exploration, for large K, and local refinement, for small K, when (simultaneously) exploring both the shape of the objective function, via the search function s given in Definition 2, and the extent of the feasible domain, via the constraint search functions $\tilde{s}_\ell$ given in Definition 3.*

2.2.2 Feasible constraint projections

The *feasible constraint projection* process (developed in §4 of [37] for linearly constrained domains), when applied to a Delaunay optimization algorithm, ensures that the maximum circumradius of the Delaunay triangulation of the datapoints remains bounded as the iterations proceed, thus leading to better-behaved uncertainty func-

tions near the boundary of the feasible domain. In this paper, since the search domain $L_s = \{x | a \leq x \leq b\}$ is a simple bound domain, implementation of the feasible constraint projection process is simpler than in [37], as described below.

We first define some preliminary concepts.

Definition 4. *A finite set of points $S^k \subset L_s = \{ x \,|\, a \leq x \leq b \}$ is called well-situated with factor of $r > 1$ if, for any point $x^k \in S^k$ and for all constraints $c^T x \leq d$ of the search domain L_s which are not active at x^k, a point $z \in S^k$ lies on the hyperplane $c^T x = d$ such that*

$$\frac{\|x^k - z\|}{\|x^k - x'\|} \leq r, \tag{2.9}$$

where x' is the projection of x^k on the hyperplane $c^T x = d$.

We now present the feasible constraint projection algorithm, developed in §4 of [37], for the simple bound domain $L_s = \{ x \,|\, a \leq x \leq b \}$.

Definition 5. *Let us consider $x^k \in L_s$, and S^k as a set in L_s that is well-situated with factor r. A feasible constraint projection is the iterative adjustment of the point x^k in L_s until the resulting augmented set, $S^{k+1} = S^k \cup \{x^k\}$, is also well situated with factor r. This projection may be achieved with Algorithm 2.1.*

Algorithm 2.1 Feasible constraint projection of x^k prior to appending to S^k.

1: If $S^{k+1} = S^k \cup \{x^k\}$ is well situated with factor r (see Definition 4), exit.
2: Otherwise, there is a constraint $c^T x = d$ that is not active at x^k, such that there is no point $z \in S^k$ that lies on the hyperplane $c^T x = d$ for which (2.9) is satisfied. In this case, redefine x^k as the projection of x^k on the hyperplane $c^T x = d$, and repeat from step 1.

In this paper, the value $r = 2$ is found to be suitable for the feasible constraint projection process in our numerical simulations. Detailed explanation of the above algorithm, and proofs of the following properties of the result, are given in [37].

Property 4. *Consider $R_{\max}$ as the maximum circumradius of a Deluanay triangulation of a set $S \subset L_s = \{x \mid a \leq x \leq b\}$ that is well-situated with factor r. Define $D = \max_{x,y \in L_s} \|x - y\|$ as the diameter of the feasible domain, and $d = \min_{1 \leq i \leq n}\{b_i - a_i\}$. Then,*

$$R_{\max} \leq D\, r_1^{n-1} \quad \textit{where} \quad r_1 = \max\{r, \frac{D}{d}\}.$$

Property 5. *Consider x' as the feasible constraint projection of x. Then,*

$$\min_{z \in S} \|z - x\| \leq \rho \min_{z \in S} \|z - x'\|, \quad \rho = [2\, r_1^2\, (1 - \frac{1}{r^2})]^{-\frac{n-1}{2}}, \quad r_1 = \max\{r, \frac{D}{d}\}.$$

That is, if the projected point x' is close to some point $z \in S$, then the original point x is correspondingly close to z as well.

Remark 2. *Since the maximum circumradius $R_{\max}$ is bounded for all iterations, the Lipschitz constant of $e(x)$ is bounded by some corresponding value L_e.*

2.2.3 A Delaunay-based constant K algorithm for solving (2.1)

A Delaunay-based constant K algorithm for solving (2.1), dubbed Δ-DOGS(Ω), is presented in Algorithm 2.2.

Figure 2.1 illustrates the application of Algorithm 2.2 to a representative $n = 2$

Algorithm 2.2 Scheme for solving (2.1) with constant K

1: Set $k = 0$ and initialize S^0 with all of the vertices of L_s together with the user-defined initial points (if any are provided). Evaluate $f(x)$ and $c_\ell(x)$, $\forall\, \ell \in \{1, 2, \cdots, m\}$, for all $x \in S^0$.

2: Calculate (or, for $k > 0$, update) interpolating functions $p^k(x)$ and $g^k_\ell(x)$ for the evaluations of $f(x)$ and $c_\ell(x)$, respectively, at all $x \in S^k$.

3: Calculate (or, for $k > 0$, update) a Delaunay triangulation Δ^k over all of the points in S^k.

4: Determine $\hat{x}^k$ as the solution of the following optimization problem:

$$\min_{x \in L_s} \; s^k(x) = p^k(x) - Ke^k(x) \tag{2.10a}$$

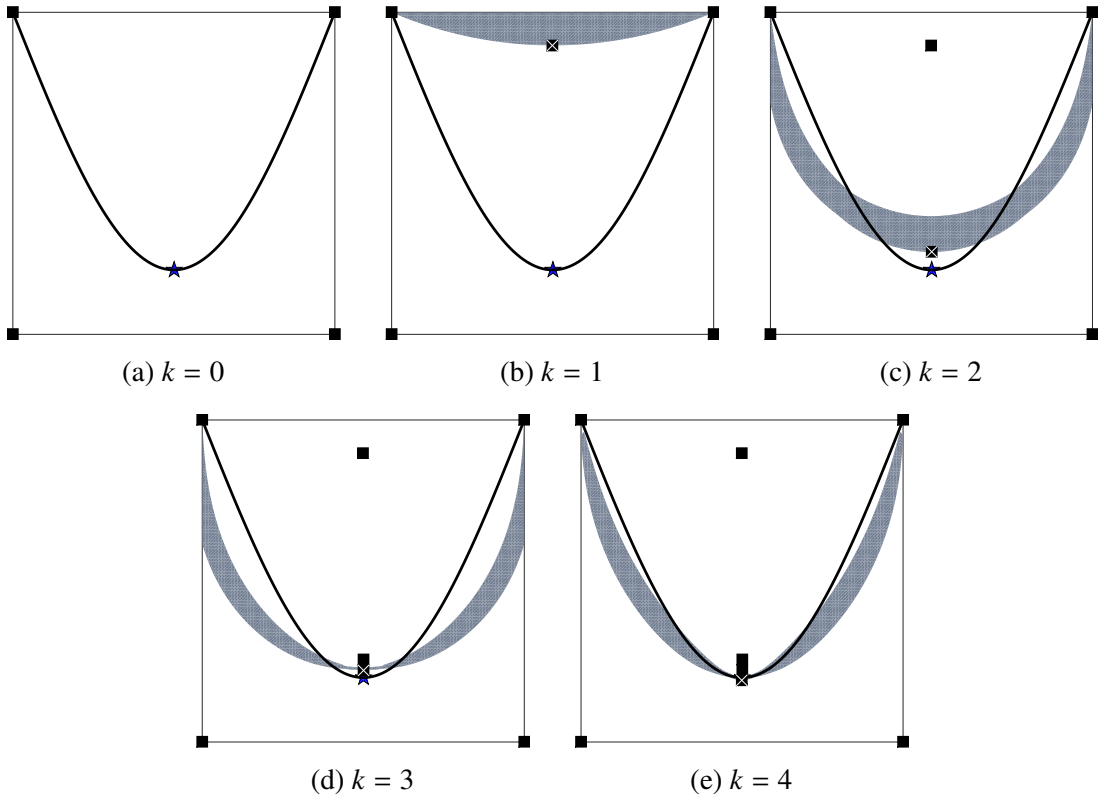
$$\text{subject to} \quad \tilde{s}^k_\ell(x) = g^k_\ell(x) - Ke^k(x) \leq 0 \quad \forall\, \ell \in \{1, 2, \cdots, m\}. \tag{2.10b}$$

where the global search function $s^k(x)$ and the ℓ'th global constraint search function $\tilde{s}^k_\ell(x)$ are introduced in Definitions 2 and 3.

5: Define x^k as the feasible constraint projection of $\hat{x}^k$ (Algorithm 2.1).

6: Evaluate $f(x^k)$ and $c_\ell(x^k)$ $\forall\, \ell \in \{1, 2, \cdots, m\}$ and set $S^{k+1} = S^k \cup \{x^k\}$. Increment k and repeat from step 2.

example, in which the objective function $f(x) = x_2$ is minimized within the search domain $L_s = \{x | 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ subject to $c(x) = x_2 + 0.8 \sin(x_1\pi) = 0$ (that is, $c_1(x) = c(x) \leq 0$ and $c_2(x) = -c(x) \leq 0$), which defines the constraint domain L_c as a 1D curve. Note that the feasible region $\Omega := L_c \cap L_s$ is nonconvex.



(a) $k = 0$ (b) $k = 1$ (c) $k = 2$

(d) $k = 3$ (e) $k = 4$

Figure 2.1: Illustration of Algorithm 2.2 on a representative example (see text). The search domain L_s is represented by the 2D square, and the global optimum is indicated by the star. The feasible domain Ω is the 1D curve shown in black.

At each iteration k, by minimizing the search function $s^k(x)$ within the approximation of the feasible domain defined by the $\tilde{s}_\ell^k(x)$ (e.g., within the shaded regions in Figure 2.1), the objective function and the extent of the feasible domain are explored simultaneously using analogous interpolating functions. The solution of (2.10) in step 4 of Algorithm 2.2 at each iteration leads to improved estimates of both the objective function and the constraint functions, and thus to the minimum of the objective function within the actual feasible domain.

Remark 3. *Algorithm 2.2 is not listed with a specific stopping criterion. It is shown in §2.3 that a limit point of the dataset obtained by this algorithm is a solution of the original problem* (2.1)*. A practical stopping criterion is easily added such that Algorithm 2.2 terminates after a finite number of iterations. A convenient stopping criterion is* $\delta^k \leq \delta_{desired}$*, where* $\delta^k = \min_{z \in S^k} \|z - x^k\|$*. It follows from the compactness of the search domain* L_s *and the Bolzano-Weierstrass theorem that such termination will occur after a finite number of iterations (see, e.g., Remark 3 in [1]). It is shown later in this paper (see Lemma 2) that a point* $z \in S^k$ *is computed via this approach such that the residual of the objective function value,* $|f(z) - f(x^*)|$*, as well as the worst-case constraint function violation,* $\max\{c_\ell(z), 0\}$ *for all* ℓ*, are both bounded by* δ^k *times a prefactor related to the Lipschitz bounds on* $f(x)$ *and* $c_\ell(x)$*, which can thus both be made as small as desired by appropriate selection of* $\delta_{desired}$*.*

Step 4 of Algorithm 2.2 computes the global minimizer of the search function within the current approximation of the feasible domain. In the next subsection, we

consider this important step in greater detail.

2.2.4 Minimizing the search function inside the approximated feasible domain

At step 4 of Algorithm 2.2, at each iteration k, the global search function $s^k(x)$ is minimized inside the approximation of the feasible domain via solution of (2.10).

Recall that, within each simplex Δ_i^k in the triangulation Δ^k, the local search function and the local constraint search functions are defined as $s_i^k(x) = p^k(x) - Ke_i^k(x)$ and $\tilde{s}_{\ell,i}^k(x) = g_\ell^k(x) - Ke_i^k(x)$, respectively. Define v_i^k as follows:

$$v_i^k = \text{argmin}_{x\in\Delta_i^k}\ s_i^k(x) \quad \text{subject to} \quad \tilde{s}_{\ell,i}^k(x) \le 0 \quad \forall\, \ell \in \{1, \cdots, m\}. \tag{2.11a}$$

The outcome x^k of step 4 of Algorithm 2.2 is thus computed as follows:

$$v^k = v_{i_{\min}}^k \quad \text{where} \quad i_{\min} = \text{argmin}_{i\in\{1,\cdots,E^k\}} \left[s_i^k(v_i^k) \right], \tag{2.11b}$$

where E^k is the number of simplices in the triangulation Δ^k. To find v^k in each simplex, we must solve E^k constrained optimization problems (2.11a), with the nonlinear constraints $\tilde{s}_{\ell,i}^k(x) \le 0$ and the linear constraints $x \in \Delta_i^k$. This computational task is simplified significantly by Lemma 1.

Lemma 1. *If the linear constraints $x \in \Delta_i^k$ in the optimization problems defined in* (2.11a) *are relaxed to the entire search domain $x \in L_s$, the resulting values of the optimal*

points remain unchanged.

Proof. Define u_j^k and u^k [cf. (2.11)] as

$$u_j^k = \text{argmin}_{x \in L_s} \; s_j^k(x) \quad \text{subject to} \quad \tilde{s}_{\ell,j}^k(x) \leq 0 \quad \forall\, \ell \in \{1, \cdots, m\}, \tag{2.12a}$$

$$u^k = u_{j_{\min}}^k \quad \text{where} \quad j_{\min} = \text{argmin}_{j \in \{1, \cdots, E^k\}} \left[s_j^k(u_j^k) \right]. \tag{2.12b}$$

We now show that u^k is also a solution of the optimization problem (2.10). By construction, $u_{j_{\min}}^k = u^k$. According to Property 3 of the uncertainty function, and the fact that $K > 0$,

$$s^k(u^k) \leq s_{j_{\min}}^k(u^k), \quad \tilde{s}_\ell^k(u^k) \leq \tilde{s}_{\ell,j_{\min}}^k(u^k) \leq 0 \quad \forall \ell \in \{1, \cdots, m\}. \tag{2.13}$$

Thus, u^k is a feasible point for optimization problem (2.10). We now check its optimality; that is, $\forall y \in L_s$ such that $\tilde{s}_\ell^k(y) \leq 0 \;\; \forall\, \ell$, that $s^k(u^k) \leq s^k(y)$. Assuming that $y \in \Delta_q^k$, by (2.12) and Property 3 of the uncertainty function,

$$\tilde{s}_{\ell,q}^k(y) = \tilde{s}_\ell^k(y) \leq 0. \tag{2.14}$$

Thus, y is a feasible point for optimization problem (2.12a). By construction of u^k, $s_{j_{\min}}^k(u^k) \leq s_q^k(u_q^k)$, and thus

$$s^k(u^k) \leq s_{j_{\min}}^k(u^k) \leq s_q^k(u_q^k) \leq s_q^k(y) = s^k(y), \tag{2.15}$$

and the optimality condition is established. □

Remark 4. *Lemma 1 shows that the (possibly, multiple) global solutions of* (2.11) *and* (2.12) *are identical. The process of solving these two problems might lead to different solutions. Note that we just need to find a global solution of* (2.11) *[or, equivalently* (2.12)*] as the algorithm proceeds, so this difference is inconsequential.*

Remark 5. *If the approximation of the feasible domain at iteration k in problem* (2.10) *is empty, then the feasible domain of the subproblem* (2.12a) *is empty for all* $j \in \{1, \cdots, E^k\}$. *In this case, using a derivative-based method like SQP, we can instead find a local minimum of the following objective function:*

$$\widetilde{\widetilde{s}}^k(x) = \sum_{\ell=1}^{m} \max\left\{\tilde{s}_\ell^k(x),\ 0\right\}. \tag{2.16}$$

At all steps that the approximation of the feasible domain is empty, x^k *is taken as the minimizer of the above function, in order to search for a feasible point.*

The solution of (2.10) can thus be obtained by solving the optimization problem (2.12a) for each $j \in \{1, \cdots, E^k\}$ (and, by Lemma 1, for $x \in L_s$). These optimization problems may be solved efficiently using standard derivative-based NLP solvers. Filter SQP [13], SNOPT [12], and IPOPT [39] are among the best derivative-based optimization algorithms available today for such nonlinear programming problems. In our implementation of Algorithm 2.2, both Filter SQP and SNOPT have been implemented. The initial point which is used when solving (2.12a) for each $j \in \{1, \cdots, E^k\}$ is taken simply as the body center of simplex j. One of the advantages of using such off-the-shelf SQP-based algorithms for this subproblem is that they can verify, at least locally,

whether or not the feasible domain of each subproblem (2.12a) is empty. If it is, then these solvers find the x that minimizes

$$\widetilde{\tilde{s}}_j^k(x) = \sum_{\ell=1}^{m} \max\left\{\tilde{s}_{\ell,j}^k(x),\, 0\right\}.$$

If all of the (2.12a) subproblems are found to be infeasible, then the resulting values of $\widetilde{\tilde{s}}_j^k(x)$ are compared in order to find the x minimizing (2.16) in the search for a feasible point, as desired, thereby ignoring the search function until a feasible region is identified.

2.2.5 Parallel implementation

The parallelization approach suggested in Algorithm 5 of [37] extends immediately to the present optimization framework in order to solve (2.1) in parallel on n_p processors. Note that the three most expensive steps of Algorithm 2.2 of the present work are as follows:

1. Evaluating the objective and constraint functions. This is assumed to be the most expensive part of the present problem; thus, a framework for simultaneously evaluating the objective and constraint functions at n_p different points of interest on a parallel computer architecture is the focus of this section.

2. Solving the optimization problem (2.12a) at each simplex via an SQP method. This part of the optimization algorithm is already "embarrassingly parallel", as each subproblem $j \in \{1, \cdots, E^k\}$ may be solved independently.

3. Partitioning the search domain into a Delaunay triangulation. An incremental method is used to update the Delaunay triangulation at each iteration, thus reducing the computational cost of this procedure somewhat. Regardless, the cost of this step increases quickly as the dimension of the problem is increased.

In Algorithm 2.2, x^k is derived by solving the optimization subproblem (2.10), then performing a feasible constraint projection. Note, however, that the uncertainty function $e^k(x)$ is independent of the interpolation function $p^k(x)$. Thus, we can calculate $e^{k+i}(x)$, for $i = 1, \ldots, n_p - 1$, before completing the objective and constraint function evaluations at x^k. That is, steps $k + i$ of Algorithm 2.2, for $i = 1, \ldots, n_p - 1$, can be performed in parallel with step k under the simplifying assumption that

$$p^{k+i}(x) = p^k(x), \quad \text{and} \quad g_\ell^{k+i}(x) = g_\ell^k(x) \quad \text{for } 1 \le \ell \le m. \tag{2.17}$$

For the purpose of parallelization, we thus impose (2.17), for $i = 1, \ldots, n_p - 1$ additional iterations, in order to determine, based on the information available at step k, the best $(n_p - 1)$ additional points to explore in parallel with x^k.

Algorithm 2.3 illustrates how this idea may be implemented for parallel computation. Note that minimizing $s^{k,i}(x)$ for $0 < i \le n_p$ is relatively easy, since $s^{k,i}(x) = s^{k,i-1}(x)$ in most of the simplices, and the incremental update of the Delaunay triangulation can be used to flag the indices of those simplices that have been changed by adding $x^{k,i}$ to $S^{k,i-1}(x)$ (see [40]).

Algorithm 2.3 Modification of Algorithm 2.2 such that, at each iteration k, n_p points are identified for parallel evaluation of the objective and constraint functions.

1: Set $k = 0$ and initialize S^0 with all of the vertices of L_s together with the user-defined initial points. Evaluate (in parallel) $f(x)$ and $c_\ell(x)$, $\forall\, \ell \in \{1, 2, \cdots, m\}$, for all $x \in S^0$.

2: Calculate (or, for $k > 0$, update) interpolating functions $p^k(x)$ and $g_\ell^k(x)$ for the evaluations of $f(x)$ and $c_\ell(x)$, respectively, at all $x \in S^k$.

3: Calculate (or, for $k > 0$, update) a Delaunay triangulation Δ^k over all of the points in S^k.

4: Determine $\hat{x}_{k,0}$, a global minimizer of $s^k(x) = p^k(x) - Ke^k(x)$ which is subject to $\tilde{s}_\ell^{k,0}(x) = g_\ell^k(x) - Ke^k(x) \leq 0$, $\forall\, \ell \in \{1, \cdots, m\}$ (as in step 4 of Algorithm 2.2). Note that this calculation may be done in parallel for each simplex. Define $x_{k,0}$ as the feasible constraint projection of $\hat{x}_{k,0}$ (Algorithm 2.1), and take $S^{k,1} = S^k \cup \{x_{k,0}\}$. Compute $\delta^{k,0} = \min_{y \in S^k} \|x_{k,0} - y\|$.

5: For each substep $i \in \{1, 2, \cdots, n_p - 1\}$, do the following:

   a. Update the Delaunay triangulation of the datapoints in $S^{k,i}$, thus defining the new uncertainty function $e^{k,i}(x)$.

   b. Determine $\hat{x}_{k,i}$ as a global minimizer of $s^{k,i}(x) = p^k(x) - Ke^{k,i}(x)$ subject to $\tilde{s}_\ell^{k,i}(x) = g_\ell^k(x) - Ke^{k,i}(x) \leq 0$, $\forall\, \ell \in \{1, \cdots, m\}$. Compute $\delta^{k,i} = \min_{y \in S^{k,i}} \|\hat{x}_{k,i} - y\|$.

   c. If $\delta^{k,i} \leq c\, \delta^{k,0}$ for some c such that $0 < c \leq 1$, replace $\hat{x}_{k,i}$ with a global minimizer of $e^{k,i}(x)$.

   d. Define $x_{k,i}$ as the feasible constraint projection of $\hat{x}_{k,i}$, and take $S^{k,i+1}(x) = S^{k,i} \cup \{x_{k,i}\}$.

6: Take $S^{k+1} = S^{k,n_p}$, and evaluate the objective and constraint functions, $f(x)$ and $c_\ell(x)$, at all of the points $\{x_{k,0}, x_{k,1}, \cdots, x_{k,n_p-1}\}$ in parallel.

7: Increment k, and repeat from step 2 until $\delta^{k,0} \leq \delta_{des}$.

2.3 Convergence Analysis

This section analyses the convergence properties of Algorithm 2.2. The analysis presented is analogous to, but somewhat different from, that presented in §4 of [37]. The convergence analysis is based on the following assumptions.

Assumption 1. *The objective and constraint functions $f(x)$ and $c_\ell(x)$, as well as the interpolating functions $p^k(x)$ and $g_\ell^k(x)$ for all k, are Lipschitz for the same Lipschitz*

constant $\bar{L}$.

Assumption 2. *There is a constant $\hat{K}$ such that, for all $x \in L_s$ and $k > 0$,*

$$\nabla^2\{f(x) - p^k(x)\} + 2\,\hat{K}I \geq 0, \quad \nabla^2\{c_\ell(x) - g_\ell^k(x))\} + 2\,\hat{K}I \geq 0,$$

$$\nabla^2 f(x) + 2\,\hat{K}I \geq 0, \quad \nabla^2 c_\ell(x) + 2\,\hat{K}I \geq 0.$$

Assumption 3. *The problem in* (2.1) *has a nonempty feasible domain, $\Omega \neq \emptyset$. Moreover, since Ω is compact, there exists a minimizer of $f(x)$ in Ω, which is denoted in this section as x^*. (In §2.5, we relax this assumption to consider the case for which the feasible domain Ω may be empty.)*

Lemma 2. *At each step of Algorithm 2.2, if $K \geq \hat{K}$, then there is a point $\tilde{x} \in L_s$ for which*

$$s^k(\tilde{x}) \leq f(x^*) \quad \textit{and} \quad \tilde{s}_\ell^k(\tilde{x}) \leq 0 \quad \textit{for } 1 \leq \ell \leq m, \tag{2.18}$$

where x^* is a global minimizer of $f(x)$ in Ω.

Proof. Consider Δ_i^k as a simplex in Δ^k which includes x^*. Define the functions $F(x)$ and $C_\ell(x)$, $\forall \ell \in \{1, \cdots, m\}$, such that

$$F(x) = p^k(x) - K\,e_i^k(x) - f(x), \quad C_\ell(x) = g_\ell^k(x) - K\,e_i^k(x) - c_\ell(x), \tag{2.19}$$

where $e_i^k(x)$ is the local uncertainty function in simplex Δ_i^k. Property 2 of the uncertainty

function states that $\nabla^2 e_i^k(x) = -2\,I$. Taking the Hessian of (2.19), we have

$$\nabla^2 F(x) = \nabla^2 p^k(x) - \nabla^2 f(x) + 2\,K\,I, \quad \nabla^2 C_\ell(x) = \nabla^2 g_\ell^k(x) - \nabla^2 c_\ell(x) + 2\,K\,I.$$

By choosing $K > \hat{K}$, according to Assumption 2, $\nabla^2 F(x)$ and $\nabla^2 C_\ell(x)$ are positive semidefinite; thus, $F(x)$ and $C_\ell(x)$ are convex inside the closed simplex Δ_i^k, which includes x^*. Thus, the maximum value of $F(x)$ is located at one of the vertices of Δ_i^k (see, e.g. Theorem 1 of [41]). Moreover, by construction, the value of $F(x)$ and $C_\ell(x)$ at the vertices Δ_i^k are zero; consequently, $F(x^*) \le 0$ and $C_\ell(x^*) \le 0$. On the other hand, $s^k(x^*) = s_i^k(x^*)$, and $\tilde{s}_\ell^k(x^*) = \tilde{s}_{\ell,i}^k(x^*)$. Therefore, $s^k(x^*) \le f(x^*)$ and $g_\ell^k(x^*) - K\,e^k(x^*) \le c_\ell(x^*) \le 0$. □

Remark 6. *Lemma 2 shows that the constrained feasible domain is nonempty if $K > \hat{K}$.*

Lemma 3. *At each step of Algorithm 2.2, if $K \ge \hat{K}$, then there is a point $z \in S^k$, such that*

$$f(z) - f(x^*) \le (\bar{L} + K\,L_e)\,\rho \min_{z \in S^k} \|z - x^k\|, \tag{2.20a}$$

$$c_\ell(z) \le (\bar{L} + K\,L_e)\,\rho \min_{z \in S^k} \|z - x^k\| \quad \forall \ell \in \{1, \cdots, m\}, \tag{2.20b}$$

where the parameter ρ is defined in Property 5, and is related to the feasible constraint projection procedure.

Proof. Choose a point $y \in S^k$ which minimizes $\delta = \min_{y \in S^k} \|y - \hat{x}^k\|$. According to

Property 1 of the uncertainty function and Assumption 1, the following inequalities hold, as in the proof of Lemma 3 in [1]:

$$|p^k(\hat{x}^k) - p^k(y)| \le \bar{L}\delta, \quad |g_\ell^k(\hat{x}^k) - g_\ell^k(y)| \le \bar{L}\delta \quad \forall \ell \in \{1, 2, \dots, m\}, \tag{2.21a}$$

$$|e^k(\hat{x}^k) - e^k(y)| \le L_e\delta. \tag{2.21b}$$

Recall that $s^k(x) = p^k(x) - K\,e^k(x)$ and $\tilde{s}_\ell^k(x) = g_\ell^k(x) - K\,e^k(x)$. Using (2.21),

$$\begin{gathered} |s^k(y) - s^k(\hat{x}^k)| \le (\bar{L} + K\,L_e)\,\delta, \quad |\tilde{s}_\ell^k(y) - \tilde{s}_\ell^k(\hat{x}^k)| \le (\bar{L} + K\,L_e)\,\delta, \\ s^k(y) = p^k(y) = f(y), \quad \tilde{s}_\ell^k(y) = g_\ell^k(y) = c_\ell(y), \\ f(y) \le s^k(\hat{x}^k) + (\bar{L} + KL_e)\,\delta, \quad c_\ell(y) \le \tilde{s}_\ell^k(\hat{x}^k) + (\bar{L} + KL_e)\,\delta. \end{gathered} \tag{2.22}$$

Since $\hat{x}^k$ is a global minimizer of $s^k(x)$ with respect to $\tilde{s}_\ell^k(x) \le 0$, it follows from Lemma 2 that $s^k(\hat{x}^k) \le f(x^*)$ and $\tilde{s}_\ell^k(\hat{x}^k) \le 0$, and thus

$$f(y) \le f(x^*) + (\bar{L} + K\,L_e)\,\delta, \quad c_\ell(y) \le (\bar{L} + K\,L_e)\,\delta. \tag{2.23}$$

In addition, $\delta \le \rho\ \min_{y \in S^k} \|y - x^k\|$ holds according to Property 5 of the feasible constraint projection. Hence, we have shown that (2.20) is true for $z = y$. □

Theorem 1. *There is an ω-limit point of the series* $\{x^k\}$ *which is a global solution of the optimization problem* (2.1).

Proof. Define $T(x) = \max\{f(x) - f(x^*), c_\ell(x)\}$. Take z^k as the value of $x \in S^k$ that minimizes $T(x)$. By construction, $T(x) \geq 0$; thus, $T(z^k) \geq 0$, and $T(z^k)$ is non-increasing with k. Since the search domain is compact, by the Bolzano-Weierstrass theorem, the series $\{x^k\}$ has an ω-limit point. Thus, for any $\varepsilon > 0$, for sufficiently large k, there are i and j such that $i < j \leq k$ and $\|x_i - x_j\| \leq \varepsilon$. Using Lemma 3 and considering that $z^k \in S^k$, we have

$$0 \leq T(z^k) \leq (2\,\bar{L} + K\,L_e)\rho\,\varepsilon$$

The above equation is true for all positive values of ε; additionally, since $T(z^k)$ is a non-increasing series; then,

$$\lim_{k\to\infty} T(z^k) = 0.$$

Now define z_1 as an ω-limit point for the z^k. By construction, $T(x)$ is a continuous function of x, which leads immediately to $T(z_1) = 0$. Thus, z_1 is a solution of (2.1). □

2.4 Adaptive K Algorithm

The tuning parameter K in Algorithm 2.2 specifies the trade-off between global exploration (for large K) and local refinement (for small K). In this section, we develop a method to adjust this tuning parameter at each iteration to maximally accelerate local refinement while still assuring convergence to the global solution of the constrained problem.

The method proposed builds on the fact that, if for each k there exists an $\tilde{x}$ such

Algorithm 2.4 Scheme for solving (2.1) with adaptive K.

This algorithm is almost identical to Algorithm 2.2. Instead of solving the subproblem (2.10) at step 4, the following search function is minimized instead:

$$s_a^k(x) = \max\left\{\frac{p^k(x) - f_0}{e^k(x)}, \frac{g_1^k(x)}{e^k(x)}, \frac{g_2^k(x)}{e^k(x)}, \ldots, \frac{g_m^k(x)}{e^k(x)}\right\}. \tag{2.25}$$

Note that, if $s_a^k(x) \leq 0$; then, the following subproblem is solved, which is equivalent to (2.10) when $K = 0$:

$$\min_{x \in L_s} \; p^k(x) \quad \text{subject to} \quad g_\ell^k(x) \leq 0 \quad \forall \ell = \{1, \cdots, m\}. \tag{2.26}$$

that $p^k(\tilde{x}) - K\, e^k(\tilde{x}) \leq f(x^*)$ subject to $g_\ell^k(\tilde{x}) - K\, e^k(\tilde{x}) \leq 0$ for all $\ell \in \{1, 2, \ldots, m\}$, then Eq. (2.18) is satisfied, which is sufficient to establish the convergence of Algorithm 2.2 in Theorem 1. It is not necessary to choose a constant value for K in Algorithm 2.2; instead, we may adapt it at each iteration k, taking K^k as bounded and nonnegative with $p^k(\tilde{x}) - K^k\, e^k(\tilde{x}) \leq f(x^*)$ subject to $g_\ell^k(\tilde{x}) - K^k\, e^k(\tilde{x}) \leq 0$ for all $\ell \in \{1, 2, \ldots, m\}$ at each iteration k.

Take f_0 as a (known) lower bound for $f(x)$ over the feasible domain Ω. By choosing K^k adaptively at each iteration of Algorithm 2.2 such that

$$0 \leq K^k \leq K_{\max}, \tag{2.24a}$$

$$\exists\, \tilde{x} \in \Omega \quad p^k(\tilde{x}) - K^k\, e(\tilde{x}) \leq f_0 \text{ and } g_\ell^k(\tilde{x}) \leq K^k\, e^k(\tilde{x}), \tag{2.24b}$$

for all $\ell \in \{1, 2, \cdots, m\}$, this variant of Algorithm 2.2, with K^k adapted at each iteration k as described above, preserves the guaranteed convergence of the original Algorithm 2.2 as established in Theorem 1.

An adaptive K variant of Algorithm 2.2, for solving (2.1) when a lower bound f_0 for the objective function $f(x)$ is available, is given in Algorithm 2.4. Note that reduced values of K^k accelerate local convergence. Thus, at each iteration k of Algorithm 2.4, we seek the smallest value of K^k which satisfies (2.24). The optimal K^k is thus taken as

$$K^k = \min_{x \in L_s} \; s_a^k(x), \tag{2.27}$$

where $s_a^k(x)$ is defined in (2.25). It is straightforward to verify that the x that minimizes (2.27) also minimizes the corresponding search function $p^k(x) - K^k\, e^k(x)$ subject to $g_\ell^k(x) \leq K^k\, e^k(x) \;\; \forall\, \ell \in \{1, 2, \cdots, m\}$. Note that, if at some iteration k the solution of (2.27) is negative, we set $K^k = 0$, and thus the search at iteration k reduces to (2.26).

Since $e^k(x)$ is defined in a piecewise fashion, to minimize $s_a^k(x)$ in L_s, we must solve several optimization problems with linear constraints. Using similar reasoning as in Lemma 1, we can relax these linear constraints.

Again, to minimize $s_{a,i}^k(x)$ within each simplex Δ_i^k, a good initial point is required. Within Δ_i^k, a minimizer of $s_{a,i}^k(x)$ generally has a large value of $e_i^k(x)$; thus, the projection of the circumcenter of Δ_i^k onto the simplex itself provides a reasonable initialization point for the search for the minimum of $s_{a,i}^k(x)$.

The minimization of $s_a^k(x)$ is a minimax problem, akin to those studied in [42, 43, 44]. In our implementation, we use the exponential fitting method to solve this minimax problem, as explained in detail in [43]. To apply this method, the gradient and Hessian of $(p^k(x) - f_0)/e^k(x)$ and $g_\ell^k(x)/e^k(x)$ are needed. Analytical expressions for

these quantities are derived in §4 in [37].

We now analyze the convergence properties of Algorithm 2.4 under the same set of assumptions as used in §2.3.

Note that the formal proof of convergence of Algorithm 2.4, given below, is not trivial. The main challenge is that the K^k derived by minimizing $s_a(x)$ is not necessarily bounded. In fact, if $f_0 < f(x^*)$, the value of K^k will go to infinity as the algorithm proceeds. Regardless, Algorithm 2.4 still converges to the global minimum, as established below.

Theorem 2. *Assuming $f_0 \leq f(x^*)$, at each iteration k of Algorithm 2.4,*

$$\min_{z \in S^k} \max\{f(z) - f(x^*), c_\ell(z)\} \leq C\,[\rho\,\delta_k + \sqrt{\rho\,\delta_k} + \sqrt[4]{\rho\delta_k}], \tag{2.28}$$

where $\delta'_k = \min_{z \in S^k} \|z - \hat{x}_k\|$, ρ is a parameter defined in Property 5 related to the feasible boundary projection process, $A = \hat{K}\,L_e + \bar{L}$, $B = (f(x^*) - f_0)\,L_e\,\bar{L}^2$, and $C = 2\,\max\{A, B, \sqrt{A}, \sqrt{B}, \sqrt[4]{A}, \sqrt[4]{B}\}$.

Proof. We first show that there is a $z \in S^k$ such that

$$\min_{z \in S^k} \max\{f(z) - f(x^*), c_\ell(z)\} \leq C\,[\delta_k + \sqrt{\delta_k} + \sqrt[4]{\delta_k}], \tag{2.29}$$

where $\delta_k = \min_{x \in S^k} \|x - \hat{x}_k\|$. By construction, there are two cases for $\hat{x}_k$.

In the first case, $\hat{x}_k$ is found by solving (2.26). By construction, $p^k(\hat{x}_k) \leq f_0$ and $g_\ell(\hat{x}_k) \leq 0$. Now take y as a point in S^k that minimizes $\|\hat{x}_k - y\|$; since $f_0 \leq f(x^*)$ and

$y \in S^k$, noting Assumptions 1-2, it follows that

$$f(y) - f(x^*) \leq \bar{L}\,\delta_k \quad \text{and} \quad c_\ell(y) \leq \bar{L}\,\delta_k, \tag{2.30}$$

which establishes that (2.29) is true in this case.

In the second case, $\hat{x}_k$ is found by solving (2.25). As $\hat{x}_k$ is the minimizer of $s_a^k(x)$, it follows that $s_a^k(\hat{x}_k) \leq s_a^k(x^*)$. There are now two possible situations for $s_a^k(x^*)$. In the first such situation, $s_a^k(x^*) = g_\ell^k(x^*) \,/\, e^k(x^*)$. Define y and z as the closest points in S^k to $\hat{x}_k$ and x^*, respectively. In Lemma 2, it is shown that $g_\ell^k(x^*) - \hat{K}\, e^k(x^*) - c_\ell(x^*) \leq 0$; thus, noting that x^* is feasible, we have

$$\hat{K} \geq g_\ell^k(x^*) \,/\, e^k(x^*). \tag{2.31}$$

Via Assumption 1 and the fact that $y \in S^k$,

$$p^k(y) = f(y), \qquad g_\ell^k(y) = c_\ell(y),$$
$$p^k(y) - p^k(\hat{x}_k) \leq \bar{L}\,\delta_k, \qquad g_\ell^k(y) - g_\ell^k(\hat{x}_k) \leq \bar{L}\,\delta_k.$$

In Lemma 2, it is also shown that $p^k(\hat{x}_k) - \hat{K}\, e^k(\hat{x}_k) - f(x^*) \leq 0$; thus, using the above equations and $g_\ell^k(\hat{x}_k) \leq \hat{K}\, e^k(\hat{x}_k)$, we have

$$f(y) - f(x^*) \leq \hat{K}\, e^k(\hat{x}_k) + \bar{L}\,\delta_k, \qquad c_\ell(y) \leq \hat{K}\, e^k(\hat{x}_k) + \bar{L}\,\delta_k.$$

By Property 1 of the uncertainty function, and the fact that $e^k(y) = 0$, it follows that $e^k(\hat{x}_k) \leq L_e\, \delta_k$, which establishes that (2.29) is true in this situation.

In the situation which is left to analyze, $s_a^k(x^*) = (p^k(x^*) - f_0)/\, e^k(x^*)$. Recall that w is the closet point to x^* in S^k. By Assumptions 1-2, and the fact that x^* is a feasible point for problem (2.1), it follows that

$$\begin{aligned} p^k(w) - p^k(\hat{x}_k) \leq \bar{L}\,\|w - \hat{x}_k\|, \qquad & g_\ell^k(y) - g_\ell^k(\hat{x}_k) \leq \bar{L}\,\|w - \hat{x}_k\|, \\ p^k(w) = f(w), \qquad & g_\ell^k(w) = c_\ell(w), \\ f(w) - f(x^*) \leq \bar{L}\,\|w - x^*\|, \qquad & c_\ell^k(w) \leq \bar{L}\,\|w - x^*\|. \end{aligned} \tag{2.32}$$

By Lemma 4 in [1], we have

$$\|w - x^*\|^2 \leq e^k(x^*). \tag{2.33}$$

Using (2.33) and the square of (2.32) leads to

$$\begin{aligned} \Big(f(w) - f(x^*)\Big)^2 \leq \bar{L}^2 e^k(x^*), \quad & c_\ell(z)^2 \leq \bar{L}^2\, e^k(x^*), \\ \max\left\{\Big(f(w) - f(x^*)\Big)^2, c_\ell(w)^2\right\} \leq \bar{L}^2\, e^k(x^*). & \end{aligned} \tag{2.34}$$

Since $\hat{x}_k$ is a minimizer of $s_a^k(\hat{x}_k) \leq s_a^k(x^*)$, and $s_a^k(x^*) = (p^k(x^*) - f_0)/\, e^k(x^*)$,

$$\frac{p^k(\hat{x}_k) - f_0}{e^k(\hat{x}_k)} \leq \frac{p^k(x^*) - f_0}{e^k(x^*)}, \qquad \frac{g_\ell^k(\hat{x}_k)}{e^k(\hat{x}_k)} \leq \frac{p^k(x^*) - f_0}{e^k(x^*)}.$$

Thus,

$$\frac{p^k(\hat{x}_k) - f_0}{e^k(\hat{x}_k)} \leq \frac{p^k(x^*) - f(x^*)}{e^k(x^*)} + \frac{f(x^*) - f_0}{e^k(x^*)}, \tag{2.35}$$

$$\frac{g^k_\ell(\hat{x}_k)}{e^k(\hat{x}_k)} \leq \frac{p^k(x^*) - f(x^*)}{e^k(x^*)} + \frac{f(x^*) - f_0}{e^k(x^*)}. \tag{2.36}$$

As in (2.31), we can show that

$$\hat{K} \geq p^k(x^*) - f(x^*) \,/\, e^k(x^*). \tag{2.37}$$

Using (2.35), (2.36), and (2.37)

$$\frac{p^k(\hat{x}_k) - f_0}{e^k(\hat{x}_k)} \leq \hat{K} + \frac{f(x^*) - f_0}{e^k(x^*)}, \qquad \frac{g^k_\ell(\hat{x}_k)}{e^k(\hat{x}_k)} \leq \hat{K} + \frac{f(x^*) - f_0}{e^k(x^*)},$$

$$\frac{p^k(\hat{x}_k) - f(x^*)}{e^k(\hat{x}_k)} \leq \hat{K} + \frac{f(x^*) - f_0}{e^k(x^*)}, \qquad \frac{g^k_\ell(\hat{x}_k)}{e^k(\hat{x}_k)} \leq \hat{K} + \frac{f(x^*) - f_0}{e^k(x^*)},$$

$$f(y) - \bar{L}\,\delta_k \leq p^k(\hat{x}_k), \qquad c_\ell(y) - \bar{L}\,\delta_k \leq g^k_\ell(\hat{x}_k),$$

$$\frac{f(y) - \bar{L}\,\delta_k - f(x^*)}{e^k(\hat{x}_k)} \leq \hat{K} + \frac{f(x^*) - f_0}{e^k(x^*)},$$

$$\frac{c_\ell(y) - \bar{L}\,\delta_k}{e^k(\hat{x}_k)} \leq \hat{K} + \frac{f(x^*) - f_0}{e^k(x^*)}.$$

We thus conclude that

$$\max\{f(y) - f(x^*), c(y)\} \leq \Big(\hat{K} + \frac{f(x^*) - f_0}{e^k(x^*)}\Big) e^k(\hat{x}_k) + \bar{L}\,\delta_k.$$

According to Property 1 and the fact that $y \in S^k$,

$$\max\{f(y) - f(x^*), c(y)\} \leq \Big(\hat{K} + \frac{f(x^*) - f_0}{e^k(x^*)}\Big) L_e\, \delta_k + \bar{L}\, \delta_k. \tag{2.38}$$

Let us define the variables ϕ_w and ϕ_y as follows:

$$\phi_w = \max\, \Big\{\Big(f(w) - f(x^*)\Big)^2, c_\ell^k(w)^2\Big\}, \quad \phi_y = \max\, \Big\{f(y) - f(x^*),\, c_\ell(y)\Big\}.$$

Using (2.34) and (2.38),

$$\phi_y\, \phi_w \leq \Big(\hat{K}\, L_e + \bar{L}\Big)\, \delta_k\, \phi_w + \Big(f(x^*) - f_0\Big)\, L_e\, \bar{L}^2\, \delta_k,$$

$$\phi_y\, \phi_w \leq A\, \delta_k\, \phi_w + \, B\, \delta_k.$$

Defining $u = \max\{\phi_y\,, \phi_w\}$ and $v = \min\{\phi_y\,, \phi_w\}$, it follows that[2]

$$v \leq 2\, \max\{A\, \delta_k,\, \sqrt{B\, \delta_k}\} \leq 2\, A\, \delta_k + 2\, \sqrt{B\, \delta_k}. \tag{2.39}$$

If $v = \phi_y$, then (2.29) is a direct outcome of (2.39); otherwise,

$$\max\, \Big\{\Big(f(w) - f(x^*)\Big), c_\ell^k(w)\Big\} \leq C\, [\, \sqrt{\delta_k} + \sqrt[4]{\delta_k}], \tag{2.40}$$

which establishes that (2.29) is true in this situation as well. Thus, (2.29) is valid for all

[2]If $A, B, C > 0$ and $A^2 \leq A\, B + C$ then $A \leq B + \, \sqrt{C} \leq 2\, \max\{B,\, \sqrt{C}\}$.

cases. By Property 5, $\delta_k \leq \rho\|x - x^k\|$; thus, (2.28) is obtained from (2.29). □

Remark 7. *By Theorem 2 above, as in Theorem 1, we can easily show that, if $f_0 \leq f(x^*)$, an ω-limit point of the datapoints determined by Algorithm 2.4 is a solution of* (2.1).

2.4.1 Using an inaccurate estimate of f_0

In the previous section, convergence of Algorithm 2.4 is proved when $f_0 \leq f(x^*)$. It is observed (see §2.6) that, if f_0 is not a tight lower bound for the global minimum, the rate of convergence is reduced. In this subsection, we study the behavior of Algorithm 2.4, when the estimated value of f_0 is somewhat *larger* than the actual minimum of the function of interest within the feasible domain. It it shown that, upon convergence, Algorithm 2.4 determines a feasible point z such that $f(z) \leq f_0$.

Theorem 3. *Assuming $f_0 > f(x^*)$, at each step of Algorithm 2.2, there is a point $z \in S^k$ such that*

$$\max\{f(z) - f_0, c_\ell(z)\} \leq [\bar{L} + \hat{K}\, L_e]\, \rho\, \delta_k, \quad \delta_k = \min_{z \in S^k} \|x^k - z\|. \tag{2.41}$$

Proof. As in Theorem 2, we first show that

$$\max\{f(y) - f_0,\ c_\ell(y)\} \leq [\bar{L} + \hat{K}\, L_e]\|y - \hat{x}_k\|, \tag{2.42}$$

where $y \in S^k$ minimizes $\delta_k = \|z - \hat{x}_k\|$. As before, during the iterations of Algorithm 2.4, there are two possible cases for step k. In the first case, $\hat{x}_k$ is found by solving (2.26).

Similar to the first case in Theorem 2,

$$p^k(y) - p^k(\hat{x}_k) \leq \bar{L}\delta_k, \quad f(y) - p^k(\hat{x}_k) \leq \bar{L}\delta_k,$$

$$g_\ell^k(y) - g_\ell^k(\hat{x}_k) \leq \bar{L}\delta_k, \quad c_\ell(y) - g_\ell^k(\hat{x}_k) \leq \bar{L}\,\delta_k,$$

$$\max\{f(y) - f_0, c_\ell(y)\} \leq \bar{L}\,\delta_k,$$

which establishes that (2.42) is true in this case.

In the second case, $\hat{x}_k$ is found by solving (2.25), and is a minimizer of $s_k^a(x)$. As in (2.31), using the fact that $f_0 > f(x^*)$, it is easy to show that

$$\max\left\{p^k(x^*) - f(x^*),\ g_\ell^k(x^*)\right\} \leq \hat{K}e^k(x^*),$$
$$s_a(x^*) = \frac{\max\{p^k(x^*) - f_0, g_\ell^k(x^*)\}}{e^k(x^*)} \leq \hat{K}.$$

Since $\hat{x}_k$ is a global minimizer of $s_a^k(x)$, it follows that

$$s_a^k(\hat{x}_k) \leq s_a^k(x^*) \leq \hat{K},$$

$$\max\left\{p^k(\hat{x}_k) - f_0,\ g_\ell^k(\hat{x}_k)\right\} \leq \hat{K}\,e^k(\hat{x}_k) \leq \hat{K}\,L_e\,\|y - \hat{x}_k\|, \tag{2.43}$$

$$p^k(y) - p^k(\hat{x}_k) \leq L_e\,\delta_k, \quad g_\ell^k(y) - g_\ell^k(\hat{x}_k) \leq L_e\,\delta_k, \tag{2.44}$$

$$p^k(y) = f(y), \quad g_\ell^k(y) = c_\ell(y). \tag{2.45}$$

Using (2.43), (2.44) and (2.45), then (2.42) is satisfied. Finally, using Lemma 2 as done in Theorem 2, (2.41) is obtained from (2.42). □

Remark 8. *Analogous to Remark 7 for the case with $f_0 \leq f(x^*)$, it follows easily from Theorem 3 above, for the case with $f_0 > f(x^*)$, that an ω-limit point of the datapoints determined by Algorithm 2.4 is a feasible point of* (2.1) *with objective function value less than or equal to f_0.*

2.5 The case of an empty feasible domain

In the previous section, it was assumed that the feasible domain Ω of the problem considered, (2.1), is nonempty. We now consider the behavior of Algorithm 2.2 when the feasible domain of (2.1) might be empty (and, thus, Assumption 9 might not hold), though Assumptions 1-2 remain in effect. We will show that Algorithm 2.2 can be used, in fact, to *verify* whether or not the feasible domain is empty.

Lemma 4. *If the feasible domain of* (2.1) *is empty, and $K > \hat{K}$; then,*

$$\sum_{\ell=1}^{m} \max\left\{c_\ell(y), 0\right\} \leq \sum_{\ell=1}^{m} \max\left\{c_\ell(x_f),\ 0\right\} + m\,(\bar{L} + K\,L_e)\,\rho \min_{z \in S^k} \|z - x^k\|, \tag{2.46}$$

where x_f is the point $x \in L_s$ that globally minimizes $\sum_{\ell=1}^{m} \max\{c_\ell(x)\,, 0\}$, and y is the point $z \in S^k$ that minimizes $\|z - \hat{x}_k\|$.

Proof. Take Δ_i^k as a simplex in Δ^k which includes x_f. In the proof of Lemma 2, it is

shown that $C_\ell(x_f) \leq 0$ if $K \geq \hat{K}$, where $C_\ell(x)$ is defined in (2.19); thus,

$$\begin{gathered} g_\ell(x_f) - K\, e^k(x_f) \leq c_\ell(x_f), \\ \sum_{l=1}^{m} \max\left\{g_\ell(x_f) - K\, e^k(x_f),\, 0\right\} \leq \sum_{\ell=1}^{m} \max\left\{c_\ell(x_f),\, 0\right\}. \end{gathered} \tag{2.47}$$

By construction, either $\hat{x}_k$ is a feasible point of (2.10), or it is a minimizer of $\sum_{\ell=1}^{m} \max\{g_\ell(x) - K\, e^k(x), 0\}$. In either case, we have:

$$\sum_{\ell=1}^{m} \max\left\{g_\ell(\hat{x}_k) - K\, e^k(\hat{x}_k),\, 0\right\} \leq \sum_{\ell=1}^{m} \max\left\{g_\ell(x_f) - K\, e^k(x_f),\, 0\right\}.$$

Using (2.47),

$$\sum_{\ell=1}^{m} \max\left\{g_\ell(\hat{x}_k),\, 0\right\} \leq m\, K\, e^k(\hat{x}_k) + \sum_{\ell=1}^{m} \max\left\{c_\ell(x_f),\, 0\right\}.$$

By construction, L_e and $\bar{L}$ are Lipschitz norms for $e^k(x)$ and $g_\ell(x)$, respectively. Moreover, $\rho\, \delta_k \leq \|y - \hat{x}_k\|$. Using Assumption (9), we have

$$\sum_{\ell=1}^{m} \max\left\{g_\ell^k(y),\, 0\right\} \leq \sum_{\ell=1}^{m} \max\left\{c_\ell(x_f),\, 0\right\} + \, m\,(\bar{L} + K\, L_e)\, \rho\, \delta_k.$$

Since $y \in S^k$, it follows that $g_\ell^k(y) = c_\ell(y)$, and thus (2.46) is satisfied. □

Remark 9. *By Lemma 4 above (cf. Theorem 1), if Algorithm 2.2 is not terminated at any step, an ω-limit point of the datapoints determined is a global minimizer of $\sum_{\ell=1}^{m} \max\{c_\ell(x), 0\}$ in L_s. If Algorithm 2.2 is terminated at step k, an approximation*

of this point is obtained; this approximation is improved as k is increased.

2.6 Results

Several benchmark optimizations are now considered to study the behaviour of the algorithms developed in this work. In §2.6.1, Algorithm 2.2 (constant K) and the (usually, more practical) Algorithm 2.4 (adaptive K) are applied to three test problems, and the roles of the tuning parameters K (on Algorithm 2.2) and f_0 (on Algorithm 2.4) are studied. In §2.6.2, the performance of Algorithm 2.3, the parallel version of Algorithm 2.2, is considered; note that Algorithm 2.4 may be parallelized in an analogous fashion. Finally, §2.6.3 compares the performance of Algorithm 2.4 with other modern derivative-free optimization methods on a representative test problem with nonconvex constraints, assuming accurate knowledge of f_0.

In the test optimizations performed in this section, polyharmonic spline interpolation [45] is used for interpolation of the known values of the objective and constraint functions. The optimizations are stopped, at iteration k, when the new datapoint, x^k, is within a $\delta_{desired}$ neighborhood of an existing datapoint (in S^k); the present simulations take $\delta_{desired} = 0.01$.

To highlight the unique features of the algorithms developed, the three test optimization problems chosen for this study, described below, have nonconvex equality and inequality constraints, in certain cases even defining disconnected feasible domains. To compare the performance of the optimization algorithms in finding a global minimum

amongst several local minima, the number of function evaluations required in order to achieve a desired level of convergence is used as the evaluation criterion.

The three test problems considered in this work are:

A) A simple linear objective function, defined over an n-dimensional space, subject to a nonlinear equality constraint, generated using a Rastrigin function $h_a(x)$, defining an $(n-1)$-dimensional nonconvex feasible domain:

$$\min_{x \in L_s} \quad f(x_1, \ldots, x_n) = x_n - 0.1, \qquad \text{subject to} \quad c(x) = x_n - h_a(x) = 0, \tag{A.1}$$

$$h_a(x) = \frac{n-1}{6} + \frac{1}{12} \sum_{i=1}^{n-1} \left\{ 3.5^2 \, (x_i - 0.7)^2 - 2 \, \cos\left(7\pi \, (x_i - 0.7)\right) \right\} + 0.1, \tag{A.2}$$

$$0 \leq x_1, x_2, \ldots, x_n \leq 1. \tag{A.3}$$

This problem has 4^{n-1} local minima, including the unique global minimum $x^* = [0.7, \cdots, 0.7, 0.1]^T$, with $f(x^*) = 0$.

B) A quadratic objective function (given by the distance to the origin), defined over an n-dimensional space, subject to a nonlinear inequality constraint, again generated using a Rastrigin function $h_2(x)$, defining an n dimensional nonconnected

feasible domain characterized by 2^n distinct "islands" within the search domain:

$$\min_{x \in L_s} \quad f(x) = x^T\, x - 0.024\, n, \qquad \text{subject to} \qquad h_b(x) \leq 0, \tag{B.1}$$

$$h_b(x) = \frac{n}{12} + \frac{1}{6} \sum_{i=1}^{n} \left\{ 4\,(x_i - 0.7)^2 - 2\,\cos\left(4\pi\,(x_i - 0.7)\right) \right\}, \tag{B.2}$$

$$0 \leq x_1, x_2, \ldots, x_n \leq 1. \tag{B.3}$$

This problem has 2^n local minima, including the unique global minimum $x^* = [0.154969, 0.154969, \ldots, 0.154969]^T$, with $f(x^*) = 0$.

C) A quadratic objective function (given by the distance to the origin), defined over a 2D space, subject to two nonlinear inequality constraints defining a nonconvex feasible domain with a "petal"-shaped hole, as proposed by Simionescu [46]:

$$\min_{x \in L_s} \quad f(x_1, x_2) = x_1^2 + x_2^2 - 0.64, \qquad \text{subject to} \quad 0 \leq h_c(x) \leq 1 \tag{C.1}$$

$$h_c(x) = x_1^2 + x_2^2 - \left\{ r_t + r_s \cos\left[n_s\, \tan^{-1}\left(\frac{x_1}{x_2}\right)\right]\right\}^2, \tag{C.2}$$

$$-1.25 \leq x_1, x_2 \leq 1.25, \tag{C.3}$$

where $r_t = 1$, $r_s = 0.2$, and $n_s = 8$. This problem has eight global minima, located at $x = [\pm 0.306, \pm 0.739]^T$ and $x = [\pm 0.739, \pm 0.306]^T$, each characterized by $f(x^*) = 0$.

Table 2.1: Implementation of Algorithms 2.2 and 2.4 on Problems A, B, and C.

Prob.	Alg.	Parameter	Conv.	Best point	# evals.
A	2.2	$K = 1$	no	$f(x^k) = 0.08$ $h_a(x^k) = -0.02$	9
		$K = 2$	yes	$f(x^k) = 0$ $h_a(x^k) = -0.002$	16
		$K = 10$		$f(x^k) = 0.007$ $h_a(x^k) = -0.077$	30
	2.4	$f_0 = 0.9 > f(x^*)$	no	$f(x^k) = 0.287$ $h_a(x^k) = 0.005$	11
		$f_0 = 0 = f(x^*)$	yes	$f(x^k) = 0.001$ $h_a(x^k) = -0.001$	18
		$f_0 = -0.02 < f(x^*)$		$f(x^k) = 0.005$ $h_a(x^k) = -0.006$	25
B	2.2	$K = 5$	no	$f(x^k) = 0.347$ $h_b(x^k) = 0.006$	11
		$K = 12$	yes	$f(x^k) = 0.002$ $h_b(x^k) = -0.015$	25
		$K = 15$		$f(x^k) = 0.002$ $h_b(x^k) = -0.015$	42
	2.4	$f_0 = 0.752 > f(x^*)$	no	$f(x^k) = 0.335$ $h_b(x^k) = -0.035$	18
		$f_0 = 0 = f(x^*)$	yes	$f(x^k) = 0.002$ $h_b(x^k) = -0.0203$	23
		$f_0 = -0.028 < f(x^*)$		$f(x^k) = -0.003$ $h_b(x^k) = 0.032$	45
C	2.2	$K = 0$	no	$f(x^k) = 0.183$ $h_c(x^k) = -0.005$	7
		$K = 5$	yes	$f(x^k) = -0.01$ $h_c(x^k) = 0.034$	16
		$K = 10$		$f(x^k) = 0.016$ $h_c(x^k) = -0.011$	30
	2.4	$f_0 = 0.16 > f(x^*)$	no	$f(x^k) = 0.152$ $h_c(x^k) = -0.133$	14
		$f_0 = 0 = f(x^*)$	yes	$f(x^k) = 0.01$ $h_c(x^k) = 0.013$	19
		$f_0 = -0.04 < f(x^*)$		$f(x^k) = -0.01$ $h_c(x^k) = 0.034$	40

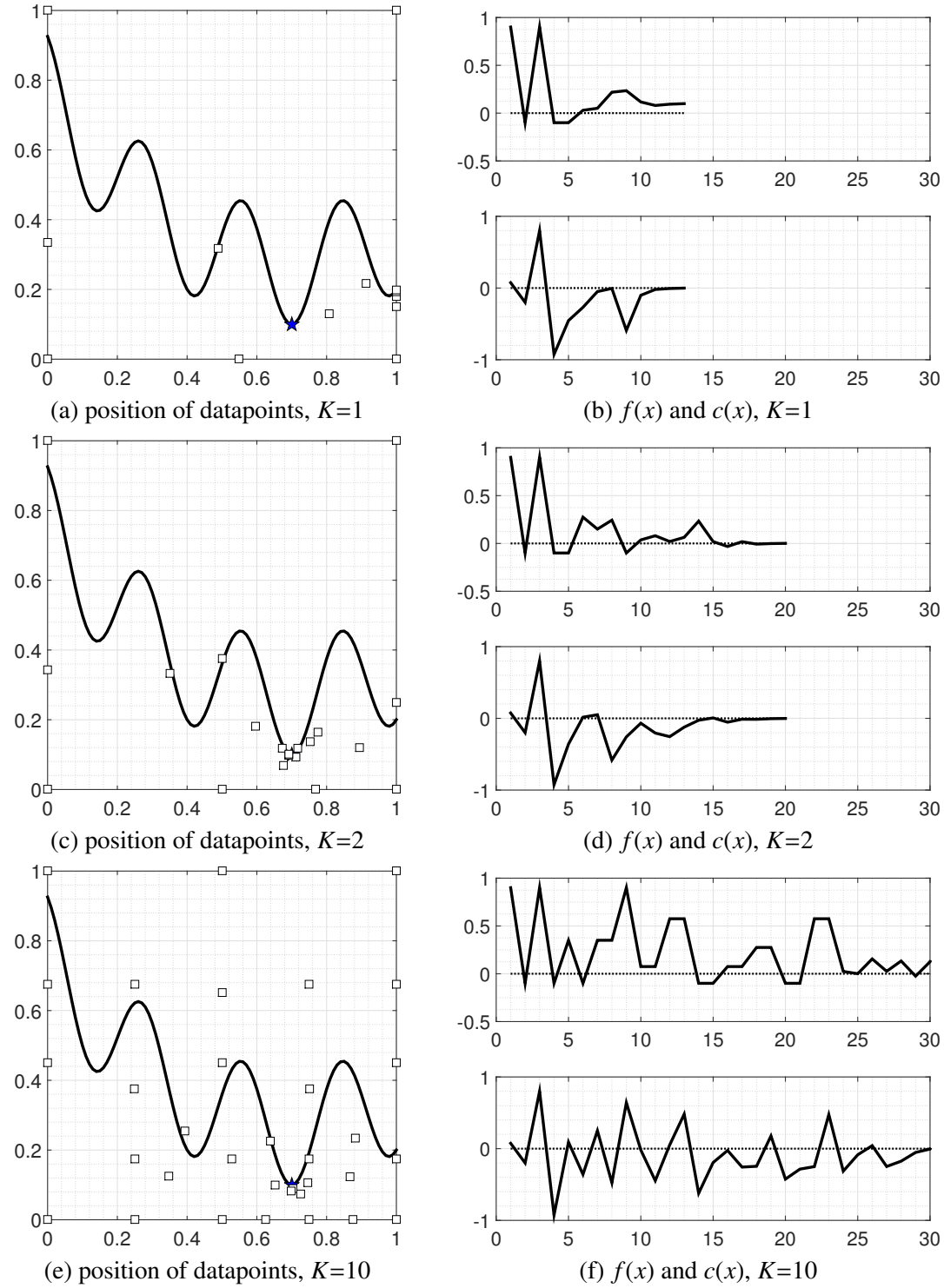
2.6.1 Performance of Algorithms 2.2 and 2.4

The performance of Algorithms 2.2 and 2.4 on Problems A, B, and C, in $n = 2$ dimensions, and the dependence of convergence on the the tuning parameters K and f_0 are compared in Table 2.1. These computations are illustrated further in the figures of this section as follows:

- Figure 2.2 illustrates the behavior of Algorithm 2.2 on Problem A,
- Figure 2.3 illustrates the behavior of Algorithm 2.4 on Problem A,
- Figure 2.4 illustrates the behavior of Algorithm 2.2 on Problem B,
- Figure 2.5 illustrates the behavior of Algorithm 2.4 on Problem B,
- Figure 2.6 illustrates the behavior of Algorithm 2.2 on Problem C, and
- Figure 2.7 illustrates the behavior of Algorithm 2.4 on Problem C.

It is observed in Table 2.1 that Algorithm 2.2 converges to the global minimum whenever the parameter K is made sufficiently large, and that unnecessarily large values of K result in additional global exploration over the search domain, slowing convergence. Similarly, it is observed that Algorithm 2.4 converges to the global minimum whenever the parameter $f_0 \leq f(x^*)$, and that unnecessarily small values of f_0 result in additional global exploration over the search domain, again slowing convergence. In cases for which $f_0 > f(x^*)$, a feasible point is identified for which objective function at least as small as f_0.

For Problem A, it is seen in Figures 2.2 and 2.3 that, even though the feasible domain in this case is a meandering 1D line over the 2D search domain, a (nearly) feasible point which (nearly) minimizes the objective function is found within a remarkably small number of function evaluations if K is taken sufficiently large in Algorithm 2.2 (or, if f_0 is taken sufficiently small in Algorithm 2.4).



(a) position of datapoints, K=1

(b) $f(x)$ and $c(x)$, K=1

(c) position of datapoints, K=2

(d) $f(x)$ and $c(x)$, K=2

(e) position of datapoints, K=10

(f) $f(x)$ and $c(x)$, K=10

Figure 2.2: Algorithm 2.2 applied to Problem A. The left plots illustrate the position of the considered datapoints and the feasible domain (dark line). The right plots illustrate the optimization history, with the optimal value for the objective function $f(x^*)$ (dashed line).

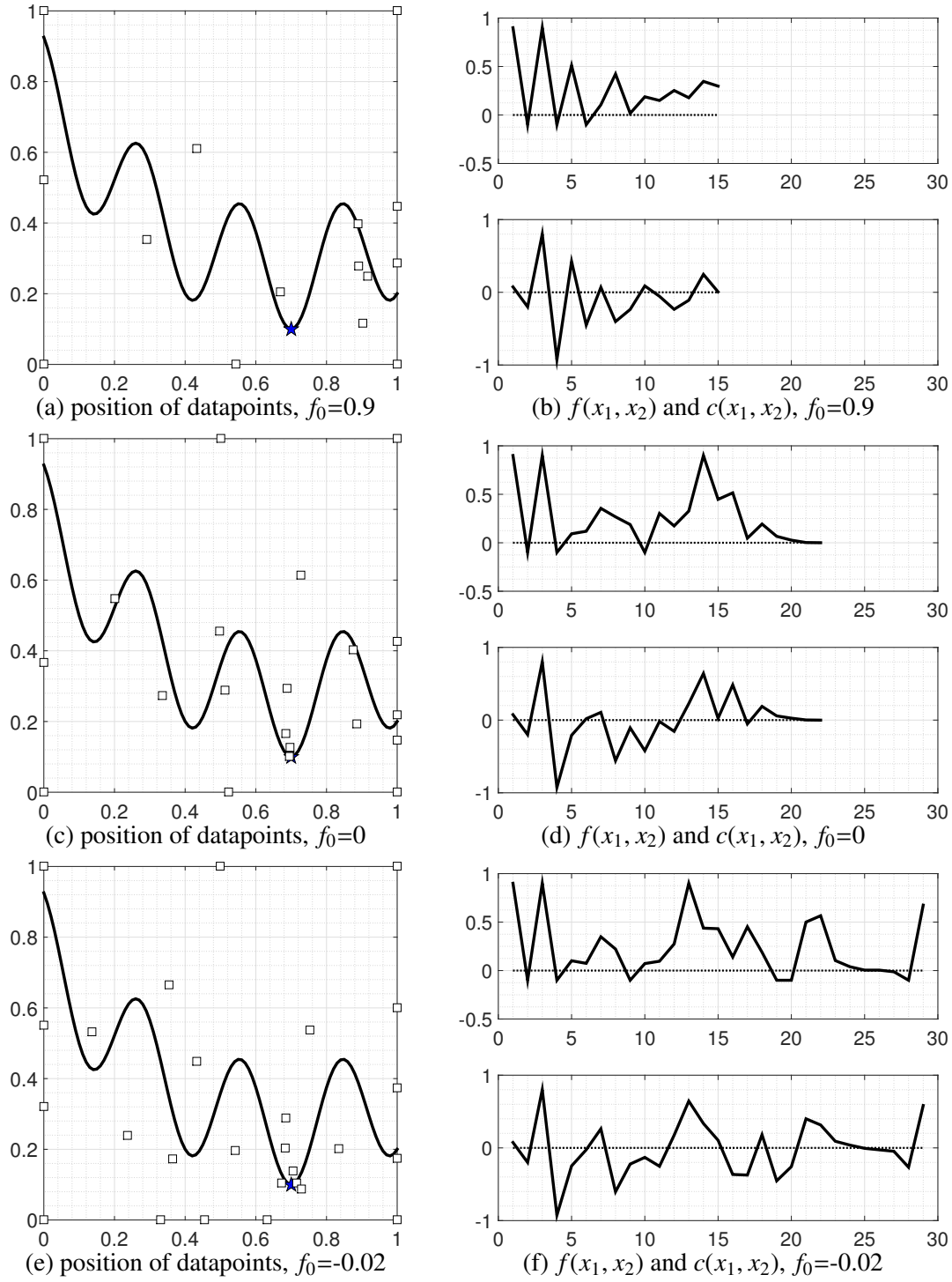
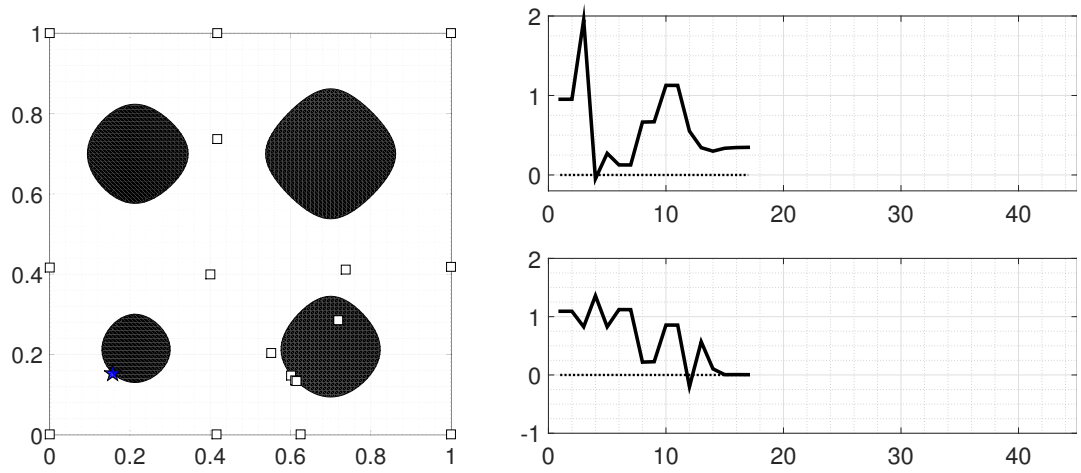
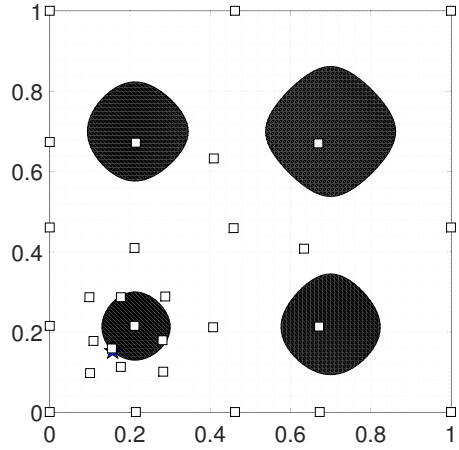


Figure 2.3: Algorithm 2.4 applied to Problem A. See caption of Fig. 2.2 for description.

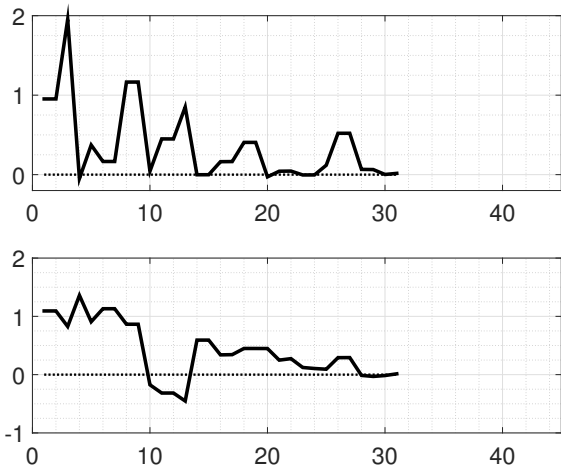


(a) position of datapoints, K=5

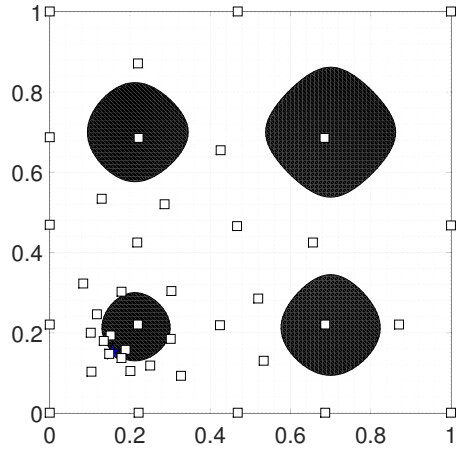
(b) $f(x_1, x_2)$ and $c(x_1, x_2)$, K=5



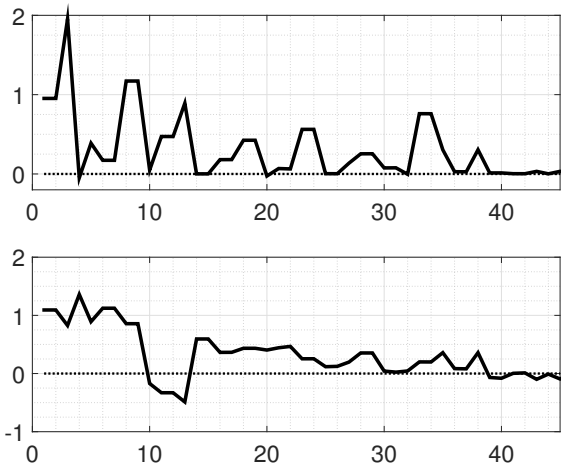
(c) position of datapoints, K=12



(d) $f(x_1, x_2)$ and $c(x_1, x_2)$, K=12



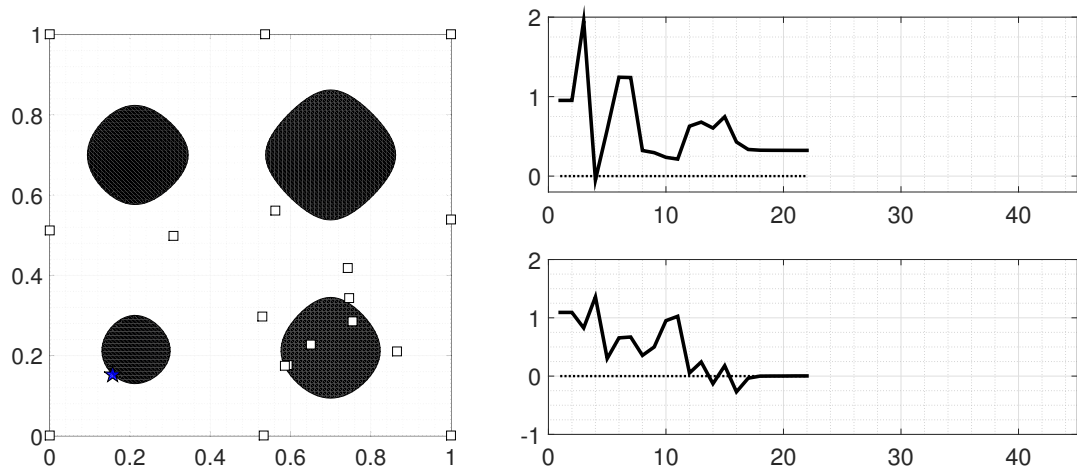
(e) position of datapoints, K=15



(f) $f(x_1, x_2)$ and $c(x_1, x_2)$, K=15

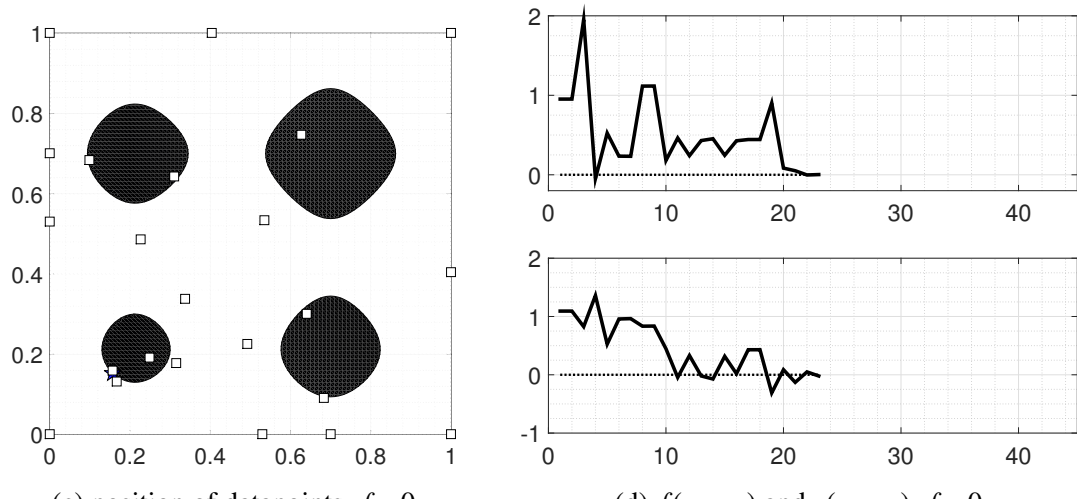
Figure 2.4: Algorithm 2.2 applied to Problem B. See caption of Fig. 2.2 for description; note that the islands of feasibility are indicated by the shaded regions in the left plots.

For Problem B, it is seen in Figures 2.4 and 2.5 that all of the islands of feasibility are sampled for K sufficiently large (for Algorithm 2.2), or for f_0 sufficiently small (for Algorithm 2.4), and that convergence is assured.



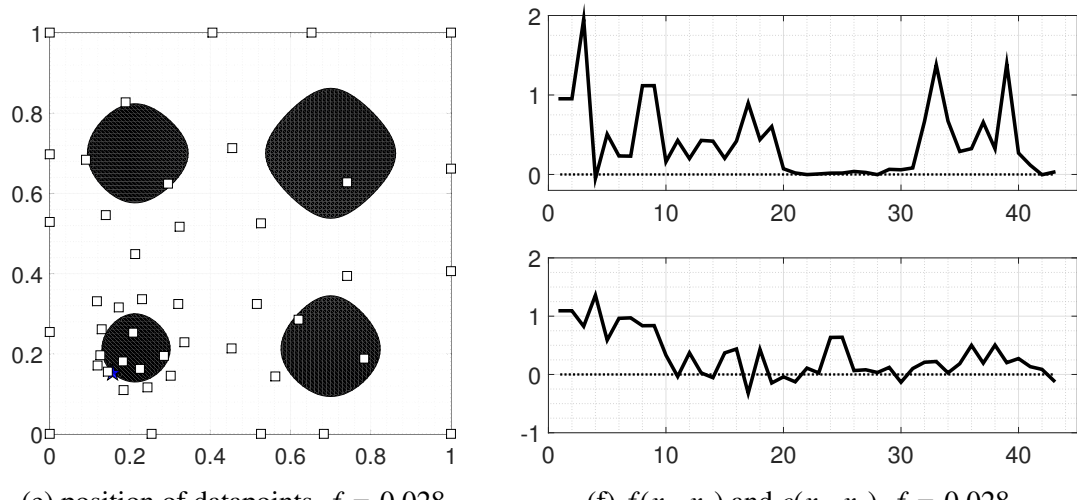
(a) position of datapoints, f_0=0.752

(b) $f(x_1, x_2)$ and $c(x_1, x_2)$, f_0=0.752



(c) position of datapoints, f_0=0

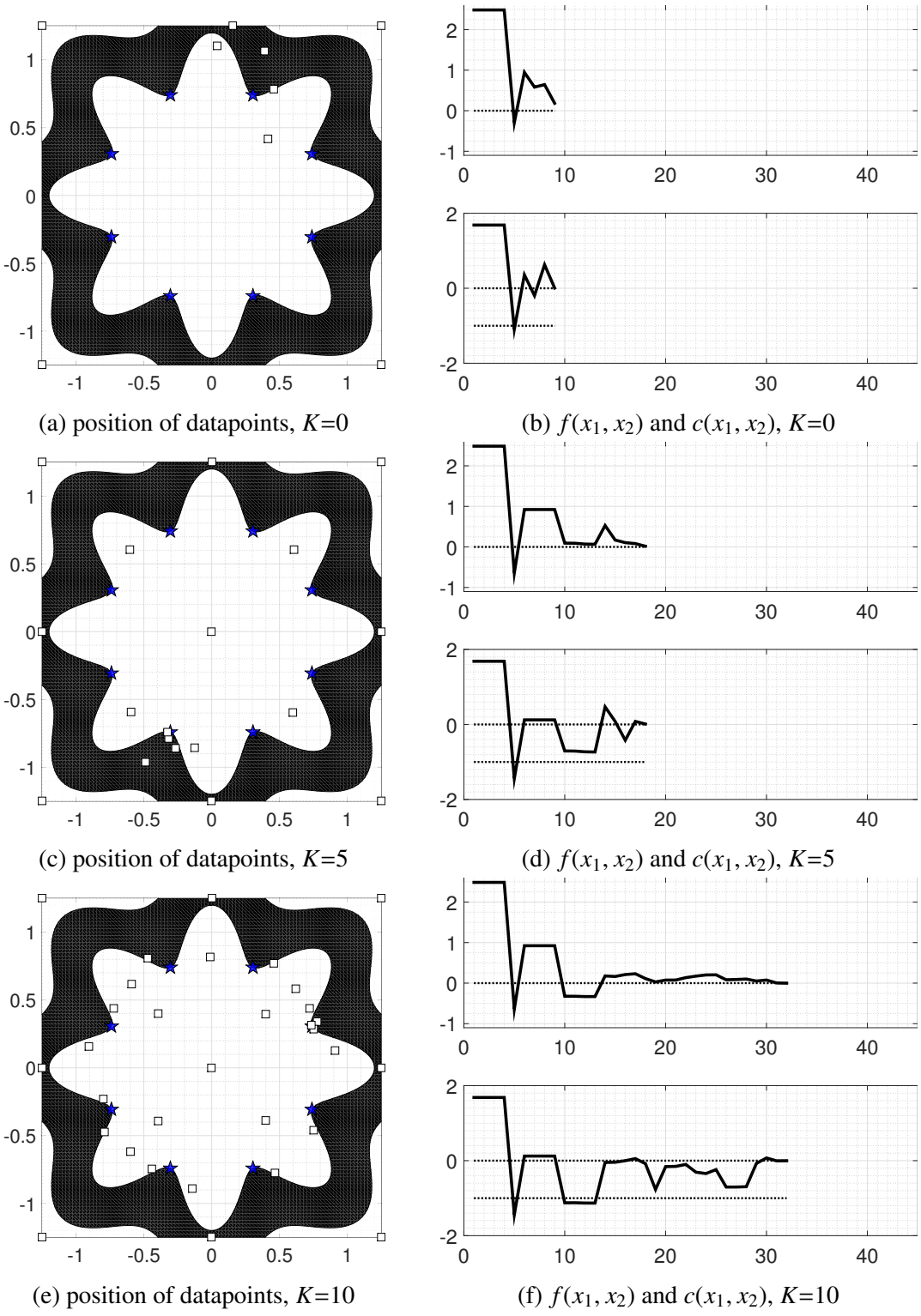
(d) $f(x_1, x_2)$ and $c(x_1, x_2)$, f_0=0



(e) position of datapoints, f_0=-0.028

(f) $f(x_1, x_2)$ and $c(x_1, x_2)$, f_0=-0.028

Figure 2.5: Algorithm 2.4 applied to Problem B. See caption of Fig. 2.4 for description.



(a) position of datapoints, K=0

(b) $f(x_1, x_2)$ and $c(x_1, x_2)$, K=0

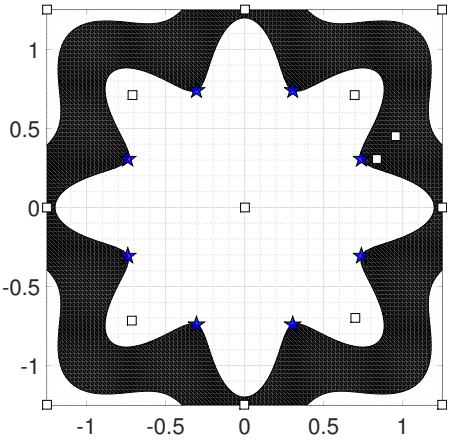
(c) position of datapoints, K=5

(d) $f(x_1, x_2)$ and $c(x_1, x_2)$, K=5

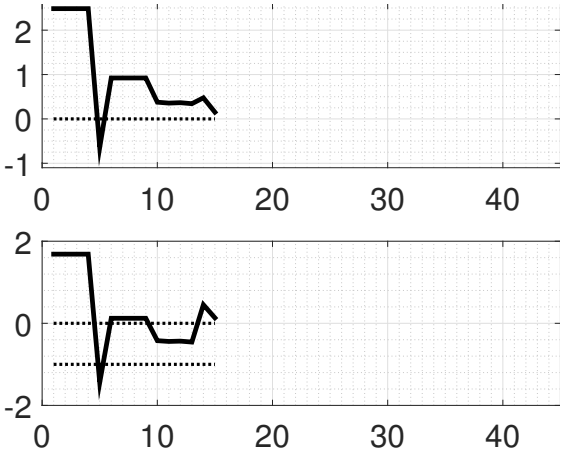
(e) position of datapoints, K=10

(f) $f(x_1, x_2)$ and $c(x_1, x_2)$, K=10

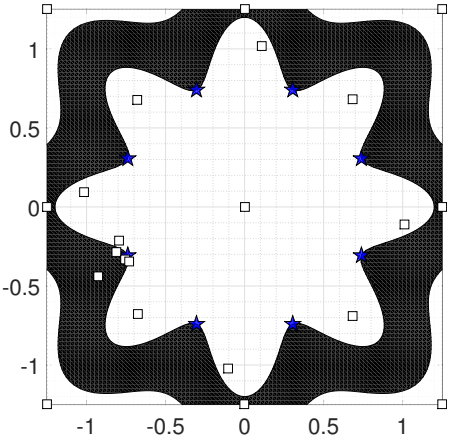
Figure 2.6: Algorithm 2.2 applied to Problem C. See caption of Fig. 2.2 for description; note that the nonconvex region of feasibility is indicated by the shaded regions in the left plots.



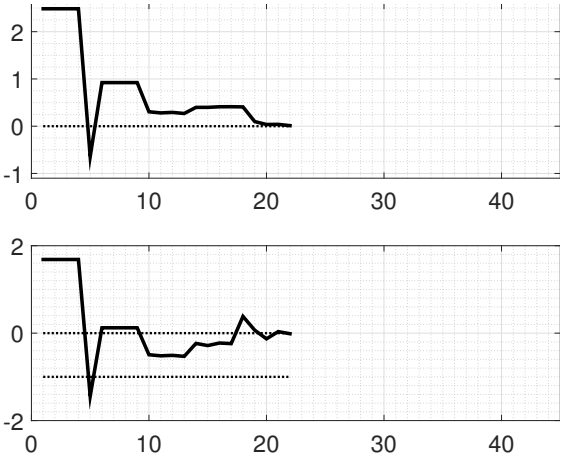
(a) position of datapoints, f_0=0.16



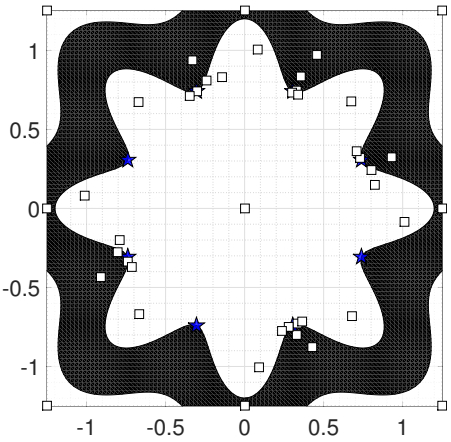
(b) $f(x_1, x_2)$ and $c(x_1, x_2)$, f_0=0.16



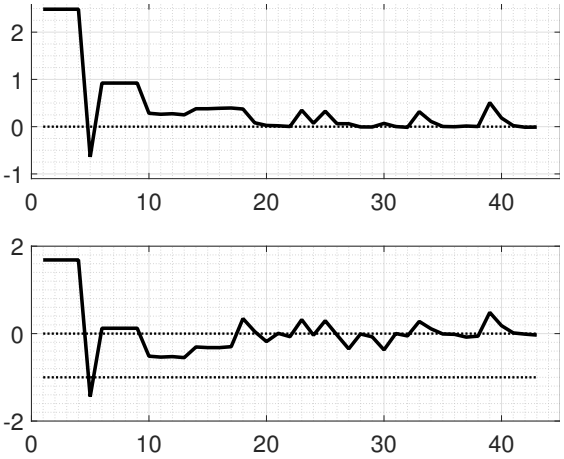
(c) position of datapoints, f_0=0



(d) $f(x_1, x_2)$ and $c(x_1, x_2)$, f_0=0



(e) position of datapoints, f_0=-0.04



(f) $f(x_1, x_2)$ and $c(x_1, x_2)$, f_0=-0.04

Figure 2.7: Algorithm 2.4 applied to Problem C. See caption of Fig. 2.6 for description.

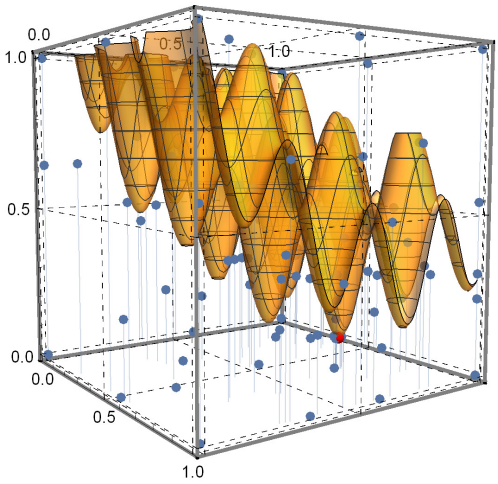
For Problem C, it is seen in Figures 2.6 and 2.7 that at least one of the eight feasible global minimum points is found if K is taken sufficiently large (or, if f_0 is taken sufficiently small), that convergence is assured, and that additional global exploration is performed if K is larger than this (or, if f_0 is taken smaller than this).

Application of Algorithm 2.4 to Problems A and B in $n = 3$ dimensions

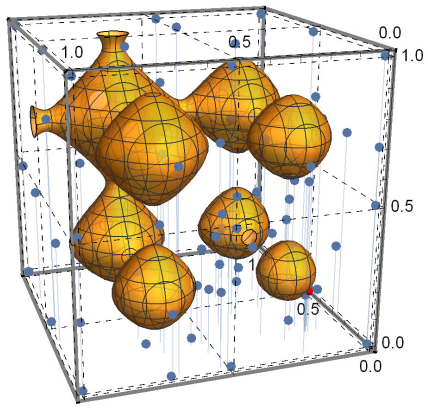
We now consider the extension of Algorithm 2.4 to Problems A and B in $n = 3$ and 4D optimization problems. Similar to the 2D cases depicted in Figures 2.2 and 2.3, we impose an equality constraint such that the feasible domain is imposed by a $n - 1$ surface, and for the second problem, an inequality constraint to generate a disconnected feasible domain; the feasible domain in the second problem is derived by an inequality constraint, so it has several disconnected regions, like multiple islands, inside the search domain.

2.6.2 Parallel performance using Algorithm 2.3

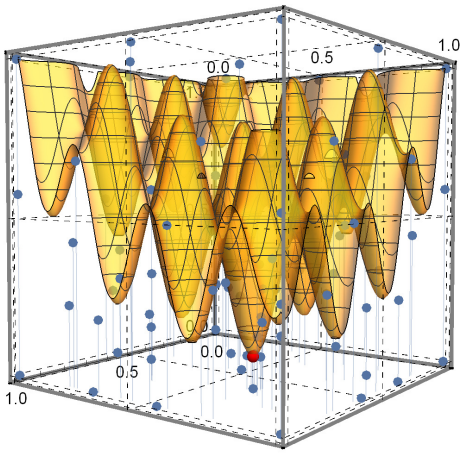
In Figure 2.9, the parallel performance of Algorithm 2.3, using $n_p = 8$ processors, is compared with the (serial) performance of Algorithm 2.2 on Problem B. The optimizations were performed with $K = 12$ in all cases, and were terminated when $\min_{x \in S^k} \|x^k - x\| \leq 0.01$. It is seen that the more frequent updates to the interpolations, as performed by the serial algorithm, do not play a very significant role in the overall number of function evaluations required by the optimization algorithm in this case. Indeed, during the first 20 function evaluations, when the search algorithm is focusing mostly



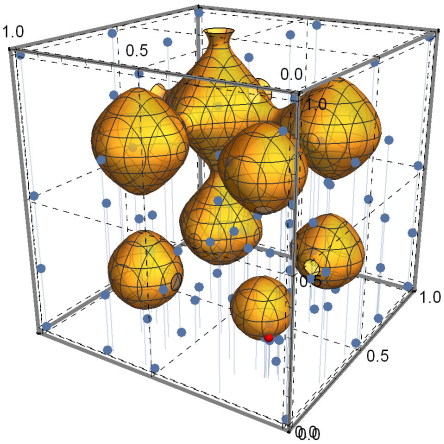
(a) Problem A



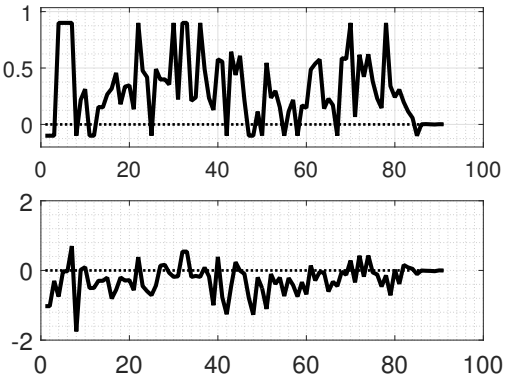
(b) Problem B



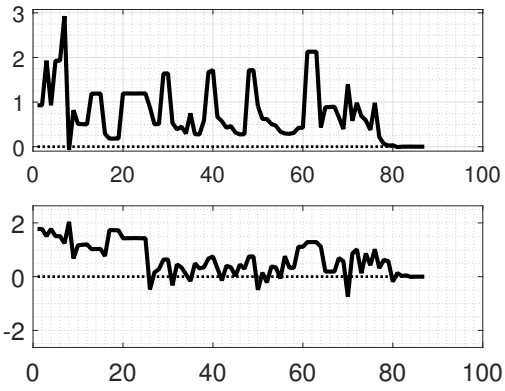
(c) Problem A



(d) Problem B



(e) $f(x^k)$ and $c(x^k)$ versus k in Problem A.



(f) $f(x^k)$ and $c(x^k)$ versus k in Problem B.

Figure 2.8: Illustration of Algorithm 2.4 on the $n = 3$ cases of Problems A and B.

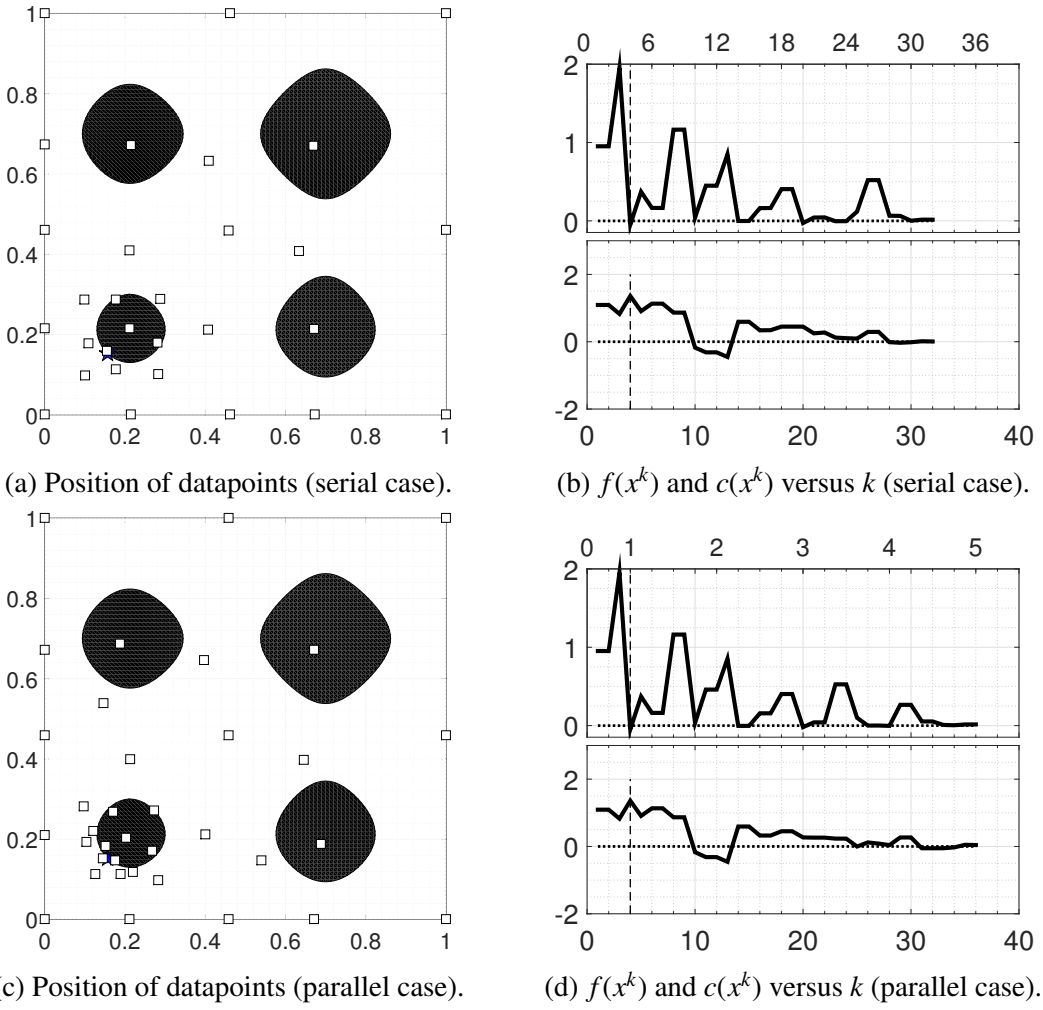
on global exploration, the function evaluations are effectively the same in the serial and parallel cases. In the later iterations, when the algorithm is focusing mostly on local refinement of the solution, the intermediate updates of the interpolant that are performed in the serial case do somewhat improve the rate of convergence when considered *per function evaluation*. However, when considering the *clock time* required, assuming a parallel computer architecture is available, the parallel algorithm is clearly superior.

In particular, assuming (as elsewhere in this paper) that the computational cost of the optimization problem considered is dominated by the cost required to perform the individual objective and constraint function evaluations, as shown in Figure 2.9, the parallel case (Algorithm 2.3) with $n_p = 8$ processors can converge to the global minimum in the clock time it takes to run only 5 function evaluations (because many functions evaluations may be performed in parallel), whereas the serial case (Algorithm 2.2) requires the clock time it takes to to run 30 function evaluations (one after the other). Thus, it is seen that the parallel implementation is able to reduce the clock time of the optimization process significantly when a parallel computer architecture is available, even though the parallel algorithm will generally perform more function evaluations.

2.6.3 Comparison with other Derivative-free methods

We now briefly compare the new optimization algorithm developed here with some other leading derivative-free optimization schemes on the difficult class of optimization problems considered in this work.

The first algorithm considered is the pattern search method called Mesh Adaptive



(a) Position of datapoints (serial case).

(b) $f(x^k)$ and $c(x^k)$ versus k (serial case).

(c) Position of datapoints (parallel case).

(d) $f(x^k)$ and $c(x^k)$ versus k (parallel case).

Figure 2.9: Comparison of (serial) Algorithm 2.2 and (parallel) Algorithm 2.9 on Problem B.

Direct Search (MADS), as proposed in [25]. We apply the efficient implementation of MADS in the NOMAD software package [47]. The NOMAD solver (implementing MADS with $2n$ neighbors) is a local derivative-free optimization algorithm that can solve difficult non-differentiable optimization problems.

The second method considered is the Surrogate Management Framework (SMF) proposed by [29]. SMF is one of the most popular derivative-free, globally-convergent, computationally-efficient optimization algorithms available today. The SMF algorithm

has two main steps, “search” and “poll”. SMF is a hybrid method that combines a pattern search with a surrogate-based optimization algorithm, and switches back and forth between (local) polling and (global) searching steps. The surrogate used by SMF is usually developed using a Kriging interpolation method. In practice, the optimization is usually initialized in the SMF approach using an initial set of datapoints generated with a Latin Hypercube Sampling (LHS) approach [29, 48, 49], which generally provides a well-distributed set of datapoints in a parameter space with box constraints, with each input variable fairly well distributed over its feasible range. The search step then uses (and updates) a Kriging-based surrogate to look for reduced objective function values inside the feasible domain, which is discretized onto a Cartesian grid that is successively refined as the iterations proceed. The search continues until it fails to return a new point on the current grid with a reduced value of the objective function, at which point a pattern search (e.g., that in MADS) is used to poll the neighborhood around the current best point. If the poll step succeeds in finding a point with a reduced objective function value, then the surrogate model is updated, and another search is performed; if it does not, the grid is refined and the process repeated. The implementation of SMF that is used in the present work is that used in [50], and was developed by our group in collaboration with Prof. Alison Marsden.

The third method considered in this section, for comparison purposes, is the Δ-DOGS(C) algorithm reported previously by our group (see [1]). Δ-DOGS(C) is a highly efficient, provably convergent, nonlinearly-constrained Delaunay-based optimization algorithm, which in many cases compares favorably with SMF in terms of computational

efficiency (see [1]). Like the present algorithm, which is also in the Δ-DOGS family, Δ-DOGS(C) leverages a synthetic uncertainty function built on the framework of a Delaunay triangulation of existing datapoints, together with an interpolation of all existing datapoints using any desired well-behaved interpolation strategy (many of our numerical experiments thus far have used polyharmonic spline interpolation).

The difficult class of problems considered in the present work is characterized by expensive constraint functions $c_\ell(x)$, as well as expensive objective functions $f(x)$; both $c_\ell(x)$ and $f(x)$ are probed on the fly as the iterations proceed, and the $c_\ell(x)$ together ultimately define a nonconvex (possibly even disconnected) feasible domain Ω. The global optimization algorithm developed in two different variants (Algorithm 2.2 with constant K, and Algorithm 2.4 with adaptive K) in the present work, dubbed Δ-DOGS(Ω), is designed specifically for problems of this class. In contrast, the three comparison methods described above [MADS, SMF, and Δ-DOGS(C)] were each designed to minimize a single nonconvex function inside a known (a priori) convex feasible domain. We thus define the following new objective function in order to apply these three existing schemes to the difficult class of problems considered in this work:

$$\tilde{f}(x) = \max\left\{f(x) - f_0, \max_{\ell=1,\ldots,m}\{c_\ell(x), 0\}\right\}. \tag{2.51}$$

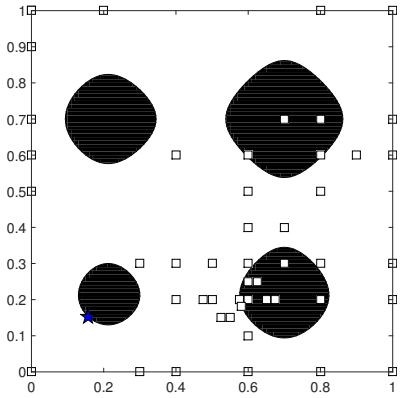
Problem B is well suited to characterize and compare the four different schemes considered here [that is, to compare MADS, SMF, and Δ-DOGS(C), with the objective function as defined in (2.51), with Δ-DOGS(Ω)] on this difficult class of problems. In

all cases, a stopping criteria of $\delta_{desired} = 0.01$ is applied [see Remark 3]. Note that many derivative-free optimization methods, like MADS, in fact do not guarantee convergence to the global minimum of the problem considered. Further, if the solution of the problem considered is on the boundary of the feasibility, as in Problem B, the task of finding a global solution is especially difficult.

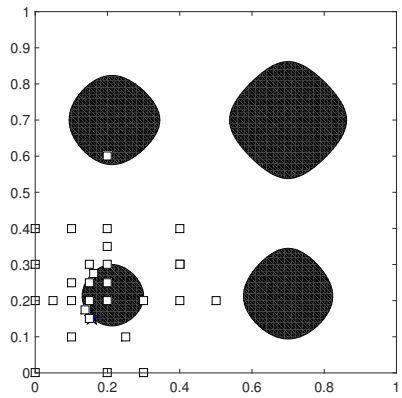
Comparison of the five plots in Figure 2.10, and the corresponding data summarized in Table 2.2, indicates that the number of function evaluations is minimized in this (typical) example using Algorithm 2.4, which accurately locates the feasible global minimizer in this case with only 23 function evaluations.

The SMF method (Figure 2.10c) explores the search domain globally, and locates the global minimizer (albeit with significantly reduced precision) with 48 function evaluations. Note that more than half of the function evaluations are performed in the polling steps, in order to guarantee convergence. Also note that the datapoints tend to accumulate in the vicinity of the feasible global minimizer as the iterations proceed, thereby causing the Kriging interpolation model to become numerically ill-conditioned. Due to this ill-conditioning of the Kriging method itself, achieving more accurate convergence using Kriging-based SMF proves to be quite difficult.

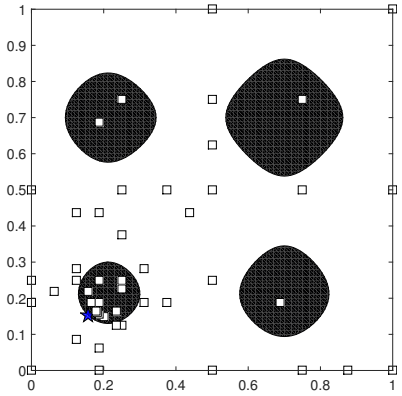
The Δ-DOGS(C) algorithm (Figure 2.10d) fails to converge to the global minimizer on this problem; since the objective function (2.51) is non-differentiable, this method is in fact not guaranteed to converge on this problem. Note, however, that this method does perform global exploration during the search, and successfully locates a feasible point near the boundary of feasibility, which is fairly near to the global mini-



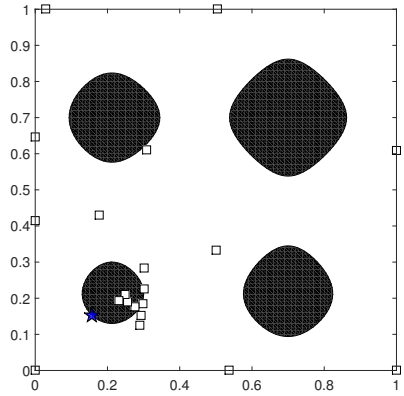
(a) NOMAD with $x_0 = [1, 0]^T$



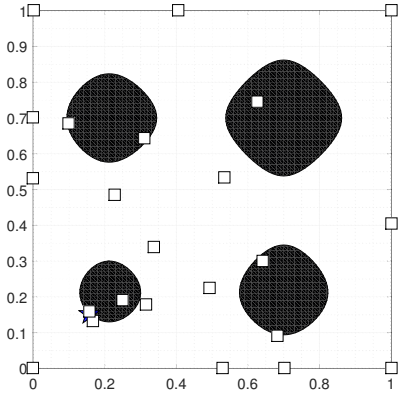
(b) NOMAD with $x_0 = [0.4, 0.3]^T$.



(c) SMF



(d) Δ-DOGS (C).



(e) Algorithm 2.4

Figure 2.10: Problem B: comparison of MADS (via NOMAD), SMF, and Δ-DOGS(C) with the adaptive K variant of the Δ-DOGS(Ω) algorithm developed and analyzed in this work, as presented in Algorithm 2.4.

Table 2.2: Performance comparison on Problem B with $n = 2$.

Algorithm	Initial point(s)	Converged	$f(z)$	$c(z)$	# of fn. evals.
MADS	$\{1, 0\}$	no	0.328	−0.011	50
MADS	$\{0.4, 0.3\}$	yes	−0.003	0.031	31
SMF	$3n + 3$ LHS points	yes	0.01	−0.074	48
Δ-DOGS(C)	$n + 1$ vertices	no	0.047	−0.147	24
Algorithm 2.4	2^n vertices	yes	0.002	−0.020	23

mizer, with only 24 function evaluations. By relaxing the stopping criterion $\delta_{desired} = 0.01$ mentioned previously, it was found that this method would eventually locate the global minimizer using about 40 function evaluations.

The MADS algorithm (Figure 2.10a-b) converges only locally. If a good initial guess is provided (Figure 2.10b), the global minimizer is located, to about the same accuracy as Algorithm 2.4, in 31 function evaluations. If a good initial guess is not provided (Figure 2.10a), the global minimizer is not located by the time the stopping criteria of $\delta_{desired} = 0.01$ is reached (after 50 function evaluations).

2.7 Conclusions

This chapter presents a new Delaunay-based derivative-free optimization approach, dubbed Δ-DOGS(Ω), of the same general family as introduced in [1, 37]. The approach developed here is designed specifically for the optimization of computationally expensive nonconvex functions within a nonconvex (possibly lower-dimensional, or even disconnected) feasible domain, which is itself defined by computationally expensive constraint functions which are explored as the iterations proceed. Two main

variants of Δ-DOGS(Ω) have been presented.

Algorithm 2.2 uses any well-behaved interpolation strategy, such as polyharmonic splines, for both the objective function $f(x)$ and the constraint functions $c_\ell(x)$, together with a synthetic piecewise-quadratic uncertainty function built on the framework of a Delaunay triangulation. A search function defined by comparing simple functions of the uncertainty model and the interpolants (of both the objective and the constraint functions) is minimized within the search domain at each iteration, and new objective and constraint function computations are performed at the optimized point, thereby refining the surrogate models of the objective and constraint functions at each iteration until convergence is achieved. Convergence to the feasible global minimum is proved mathematically under reasonable technical conditions on the smoothness of the objective and constraint functions.

Algorithm 2.4 modifies Algorithm 2.2 to use an estimate of the lower bound of the function in order to maximally accelerate local refinement while still ensuring convergence to the global minimizer.

We have also proposed, in Algorithm 2.3, a framework to efficiently parallelize on multiple processors the objective and constraint function evaluations required by Algorithm 2.2 at each step in the optimization process; this parallelization approach extends in an obvious fashion to the parallelization of Algorithm 2.4.

There is an inherent "curse of dimensionality" associated with derivative-free optimization problems. High-dimensional derivative-free optimization problems are generally computationally intractable with any method; for high-dimensional optimization

problems, derivative-based methods should always be preferred.

The main limitation of the derivative-free optimization algorithms developed in the present work, and the related optimization algorithms in [1, 37], is the memory requirements of the Delaunay triangulation algorithms upon which this body of work is based. The required Delaunay triangulations makes the "curse of dimensionality" associated with this family of optimization algorithms even more pronounced than it might be otherwise. Constructing Delaunay triangulations in problems higher than about ten dimensions is generally computationally intractable. The focus in this body of work on the use of Delaunay triangulations, which are generally the most "regular" triangulations possible for a given distribution of datapoints, provides an essential ingredient in our proofs of convergence. Regardless, for practical applications, the notion of using triangulations that are "nearly" Delaunay might be helpful in future work, for practically extending the algorithms developed here to somewhat higher-dimensional problems.

Though the test problems considered in this paper illustrate well the key features of the new optimization algorithms presented here, in future work we will test these optimization algorithms on additional benchmark and application-focused problems. We will also consider different interpolation approaches, as alternatives to the standard polyharmonic spline interpolation approach used here.

Problems in which the feasibility at point x is only computable in a binary fashion (feasible or infeasible), rather than given by the union of inequalities based on computable (and, smooth) constraint functions $c_\ell(x)$, will also be considered. Problems in which both the objective and constraint function evaluations are inaccurate will also be

considered.

Acknowledgements

Chapter 2 contains material that has been submitted for publication and is under review. S. R. Alimo, P. Beyhaghi, T. R. Bewley, “Deluanay-based optimization algorithm via global surrogates: non convex constraints ", *Journal of Global Optimization*. The dissertation author was the primary researcher and author of this material.

Chapter 3

Design of IMEXRK time integration schemes via Delaunay-based derivative-free optimization with nonconvex constraints and grid-based acceleration

Be happy for this moment. This moment is your life.—Omar Khayyam

3.1 Introduction

Over the past 30 years, the Direct Numerical Simulation (DNS) and Large Eddy Simulation (LES) of turbulent flows have been based on a wide variety of different strategies for discretization of the spatial derivatives of the governing Navier-Stokes equation, including pseudospectral representations, finite differences, Padé methods, finite elements, spectral elements, and wavelet-based approaches. Surprisingly, however, much less attention has been applied to the exploration of suitable time-marching approaches for such problems. Early work, in [51, 52], combined an implicit second-order Crank-Nicolson (CN) scheme for the time integration of the stiff terms with an explicit second-order Adams-Bashforth (AB2) scheme for the time integration of the non-stiff terms; this implicit/explicit (IMEX) combination is commonly called a CNAB scheme. This approach was later refined in [53], in which the explicit component was replaced by the third-order low-storage Runge-Kutta-Wray (RKW3) scheme [54]; this IMEX combination, dubbed CN/RKW3, is an example of a broad class of time-marching schemes commonly known as IMEXRK schemes. The CN/RKW3 scheme is only second-order accurate overall, and has an implicit component which is only *A*-stable. The CN/RKW3 scheme was improved somewhat in [55], in which a scheme dubbed here as IMEXRKSMR2 was introduced. This scheme applies the same three-step incremental formulation of CN/RKW3, with a modified implicit component which improves its accuracy and stability properties. Though the implicit component of the resulting method was made strongly *A* stable by this effort, it is not possible, as noted by [55], to

achieve full third-order accuracy with a scheme of this specific form.

The *A*-stable CN/RKW3 and strongly *A*-stable IMEXRKSMR2 schemes mentioned above, both of which are second-order accurate, facilitate low-storage implementations in which only two or three "registers" per system state in the spatially-discretized system need to be stored in memory in order to march the system considered in time. These schemes have thus dominated the turbulence simulation literature since their introduction long ago.

Recently, in [4], it was shown that much can be gained by either relaxing the low-storage requirement, allowing an additional register to be used by the time-marching algorithm, or increasing the number of stages used to perform the time advancement over each timestep. In particular, by generalizing and extending the incremental formulation used by CN/RKW3 and IMEXRKSMR2 to a four-step scheme, it was possible to achieve full-third order accuracy and better stability properties for both the implicit and the explicit parts of the scheme. The resulting scheme was obtained by imposing the accuracy constraints in a manner that reduced the parameter space to be searched to a three-dimensional bounded domain. An extensive and time-consuming brute-force search was then performed over this bounded 3D domain in order to optimize the scheme's fourth-order truncation error and stability properties.

As mentioned in the abstract, the optimization of the parameters of IMEXRK schemes involves

1. a complicated set of nonconvex constraints, which are imposed to achieve the de-

sired order of accuracy and a handful of important stability properties, and which leads to a challenging disconnected feasible domain, and

2. a highly nonconvex objective function, which represents a compromise between a few different measures characterizing the leading-order error and the potential stability shortcomings of the resulting scheme.

This structure makes the computation of new IMEXRK schemes a challenging practical test problem for global optimization algorithms. Indeed, most available optimization algorithms are ill-suited for problems of this structure. As a result, these problems are generally approached with expensive direct search methods [56].

The present work shows that such tedious and time-consuming manual searches can be automated and remarkably accelerated leveraging appropriately-designed, globally-convergent optimization algorithms.

Formulating the challenge of finding appropriate coefficients for a four-step low-storage incremental Runge-Kutta scheme as an optimization problem leads to a difficult nonlinear programming problem with multiple local minima and several nonlinear constraints. Derivative-based optimization methods applied to such problems often get stalled at local minima, failing to explore parameter space thoroughly enough to locate the global minimum [57].

Derivative-free optimization methods, on the other hand, are often designed from the outset to achieve global convergence. Unfortunately, it is difficult with many such methods to handle general nonlinear constraints. Although several efforts have appeared

in the literature to handle linear and convex constraints within the surrogate-based [1, 37] and pattern-search [26] frameworks, to the best of our knowledge, only the algorithm in [58] guarantees global convergence under the assumptions of smooth nonconvex constraints and a smooth nonconvex cost function. For this reason, the algorithm in [58] is the starting point for the present chapter.

Delaunay-based optimization algorithms [1, 37, 58] represent a new, computationally efficient, highly extensible class of derivative-free optimization methods. They have been developed to address a range of practical nonconvex optimization problems whose function evaluations are computationally (or, experimentally) expensive. These new algorithms, which are provably globally convergent under the appropriate assumptions, are response surface methods which iteratively minimize metrics based on an interpolation of existing datapoints and a synthetic model of the uncertainty of this interpolant, which itself is built on the framework of a Delaunay triangulation over the existing datapoints. Unlike other response surface methods, these algorithms are designed to leverage any well-behaved interpolation strategy.

There are four main algorithms developed in this class thus far, which address a wide range of practical optimization problems. The first [37], dubbed Δ-DOGS, efficiently minimizes expensive objective functions inside linearly constrained feasible domains. The second [1] extends Δ-DOGS to handle efficiently more general convex search domains. The third [58] extends Δ-DOGS to nonconvex and numerically expensive constraint functions. The fourth [59] incorporates a grid into Δ-DOGS to achieve faster convergence, primarily by performing fewer function evaluations along

the boundary of feasibility.

A key limitation of the algorithm developed in [58], dubbed Δ-DOGS(Ω), is the overexploration over the boundary of the search domain, L_s, that might otherwise be unnecessary. This characteristic is caused in [58] by the poor behaviour of the generated uncertainty function near the boundary of search domain. This behaviour is exacerbated when the objective function itself has irregular behavior close to the boundary of L_s, which is especially common in situations in which the objective function value on the boundary is close to the minimum.

To address this limitation, we introduce in this chapter a new variant of Δ-DOGS that significantly accelerates the algorithm developed in [58], which optimizes nonconvex and computationally expensive functions within nonconvex feasible domains, by incorporating a Cartesian grid, as motivated by [59], to significantly reduce the number of function evaluations performed on the boundary of the search domain.

This chapter is organized as follows. Section 3.2 describes our new (accelerated) optimization algorithm for solving nonconvex, computationally expensive optimization problems within nonconvex feasible domains. Section 3.3 analyzes the convergence of this algorithm. Section 3.4 outlines the accuracy constraints that need to be satisfied for an incremental IMEXRK scheme to be third-order accurate; stability properties and other metrics are also introduced. Section 3.5 provides a detailed description of how the new optimization algorithm is used to design the new IMEXRK scheme. Section 3.6 presents numerical results, demonstrating the third-order accuracy of the IMEXRK scheme determined. Conclusions and future directions are discussed in Section 3.7.

3.2 The Optimization Algorithm

In this section, we modify Δ-DOGS(Ω) to obtain faster convergence. The algorithm developed is a globally-convergent derivative-free optimization method designed to solve the following problem:

$$\text{minimize } f(x) \quad \text{with} \quad x \in \Omega := L_c \cap L_s \subseteq \mathbb{R}^n \quad \text{where}$$

$$L_c = \{x | c_\ell(x) \leq 0\,, \text{ for } \ell = 1, \cdots, m\}, \quad L_s = \{x | a \leq x \leq b\}, \tag{3.1}$$

where both $f(x)$ and $c_\ell(x)$ for $\ell = 1, \cdots, m$ are twice differentiable and possibly nonconvex functions which map $\mathbb{R}^n \to \mathbb{R}$ within the search domain L_s.

The optimization problem (3.1) has two sets of constraints:

a. a set of $2\,n$ bound constraints that characterize the n-dimensional box domain $L_s = \{x | a \leq x \leq b\}$, dubbed the *search domain*, and

b. a set of m possibly nonlinear inequality constraints $c_\ell(x) \leq 0$ that together characterize the possibly nonconvex domain $L_c = \{x | c_\ell(x) \leq 0\,, \text{ for } \ell = 1, \cdots, m\}$, designated the *constraint domain*.

The *feasible domain*, Ω, is the intersection of these two domains, $\Omega := L_s \cap L_c$.

3.2.1 Preliminary definitions

Let us first present some preliminary definitions.

Definition 6. *Given S as a set of points that includes the vertices of domain L_s, and Δ as a Delaunay triangulation of S, we define the* local uncertainty function*, $e_i(x)$, for each simplex $\Delta_i \in \Delta$*

$$e_i(x) = r_i^2 - \|x - Z_i\|^2, \tag{3.2}$$

where r_i and Z_i are the circumradius and circumcenter of Δ_i. The global uncertainty function*, $e(x)$, is*

$$e(x) = e_i(x), \quad \text{for all } x \in \Delta_i. \tag{3.3}$$

The uncertainty function (3.3) *has the following properties*[1]*:*

1. *For all $x \in L_s$, $e(x) \geq 0$. For all $x \in S$, $e(x) = 0$.*

2. *The piecewise quadratic uncertainty function* (3.3) *is continuous and Lipschitz.*

3. *For any $x \in L_s$, the uncertainty function $e(x)$ is equal to the maximum of the local uncertainty functions $e_i(x)$:*

$$e(x) = \max_i e_i(x) \quad \text{for all } x \in L_s. \tag{3.4}$$

In this work, we assume that there is a known target value f_0 that is achievable (that is, $\exists\, x \in L_s$ such that $f(x) \leq f_0$ and $c_\ell(x) \leq 0$ for all $\ell = 1, \ldots, m$). The $c_\ell(x)$ are nonlinear, computationally expensive constraint functions. Note that the present algorithm can be easily extended to problems for which a target value and constraint

[1] Proofs of these properties are provided in §3 of [37].

violation thresholds are not available, as in Algorithm 1 of [58].

Definition 7. *Take S_E^k as the set of evaluated points at iteration k, which includes the vertices of domain L_s, at which the objective and constraint functions $f(x), c_1(x), \cdots, c_m(x)$ have been evaluated. Define $p^k(x)$, $g_1^k(x), \cdots, g_m^k(x)$ as smooth interpolations of the objective and constraint functions, respectively, through all points in S_E^k. Define*

$$F^k(x) = \max\left[\ p^k(x) - f_0,\ g_1^k(x), \cdots,\ g_m^k(x)\ \right]. \tag{3.5}$$

The **continuous search function** *is defined, $\forall x \in L_s$ such that $x \notin S_E$, as*

$$s_c^k(x) = \begin{cases} F^k(x)/e^k(x), & \text{if } F^k(x) \geq 0, \\ F^k(x), & \text{otherwise}. \end{cases} \tag{3.6}$$

The **discrete search function** *is defined, $\forall x \in L_s$ such that $x \notin S_E$, as*

$$s_d^k(x) = \begin{cases} F^k(x)/Dist\{x, S_E^k\}, & \text{if } F^k(x) \geq 0, \\ F^k(x), & \text{otherwise}, \end{cases} \tag{3.7}$$

where $Dist\{x, S_E^k\}$ is defined as the minimum distance between the point x and the set of points in S_E^k:

$$Dis\{x, S_E^k\} = \min\{\|x - z\| \,|\, z \in S_E^k\}.$$

3.2.2 Summary of the original Δ-DOGS(Ω) algorithm

Delaunay-based Derivative-free Optimization via Global Surrogates (Δ-DOGS) is a generalizable family of practical, efficient, and provably-convergent derivative-free algorithms designed for a range of nonconvex optimization problems with expensive function evaluations [1, 37]. A new variant in this family of algorithms, dubbed Δ-DOGS(Ω), was developed in [58] to solve optimization problems with both nonconvex and computationally expensive objective functions *and* nonconvex and computationally expensive constraint functions, of the form (3.1). The feasible domain Ω defined in such a problem may be nonconvex; indeed, it may even be disconnected.

In the original Δ-DOGS(Ω) algorithm, approximations $p^k(x)$ and $g_\ell^k(x)$ of, respectively, the objective function $f(x)$ and the constraint functions $c_\ell(x)$ are developed at each iteration k, based on the data obtained thus far, and refined as the iterations proceed. No prior knowledge of the constraint functions is assumed. The solution of (3.1) [that is, minimization of $f(x)$ subject to $c_\ell(x) \leq 0$ and $x \in L_s$] is found iteratively, leveraging at each iteration k the continuous search function $s_c^k(x)$ defined in (3.6), which is based upon these approximations, the uncertainty function $e^k(x)$ defined in (3.3) [which for any x and k quantifies our confidence in these approximations], and a target value f_0 for the objective function $f(x)$. The algorithm is initialized by evaluating $f(x)$ and $c_\ell(x)$ at all vertices of the search domain L_s.

The essential steps of Δ-DOGS(Ω) are summarized in Algorithm 3.1. Note that any smooth interpolation strategy can be used in Algorithm 3.1; a common choice for

Algorithm 3.1 Δ-DOGS(Ω), designed for minimization of (3.1), from [58].

1: Set $k = 0$. Take S^0 as the initial set of datapoints, which are the vertices of the search domain L_s. Evaluate $f(x)$ and $c_\ell(x)$ for all $\ell \in \{1, \cdots, m\}$ at S^0.
2: Calculate interpolating functions $g_\ell^k(x)$ for the evaluations of $c_\ell(x)$, and $p(x)$ for the evaluations of $f(x)$, over all points in S^k (see §4 of [58] for details).
3: Perform a Delaunay triangulation, Δ^k, over all points in S^k.
4: For each simplex Δ_j^k of the triangulation Δ^k, calculate z_j^k and r_j^k as the circumcenter and circumradius of Δ_j^k.
5: Noting the definitions of $e^k(x)$ in (3.3) and $s_c^k(x)$ in (3.6), determine x_k as a global minimizer as follows

$$x_k = \min_{x \in L_s} s_c^k(x) \quad \text{subject to } x \in \Omega. \tag{3.8}$$

6: Evaluate $f(x_k)$ and $c_\ell(x_k)$, set $S^{k+1} = S^k \cup \{x_k\}$, and repeat from step 2 until convergence.

the surrogate models $g_\ell^k(x)$ and $p(x)$ is polyharmonic-spline interpolation [9, 60, 37].

3.2.3 Δ-DOGS(Ω_Z): implementation of Cartesian grids to accelerate Δ-DOGS(Ω)

We now present the modified optimization algorithm proposed in the present work. One of the drawbacks of Δ-DOGS(Ω), as summarized in Algorithm 3.1, is its overexploration of the boundaries of feasibility; we thus now incorporate a grid to improve its convergence properties. The new algorithm, dubbed Δ-DOGS(Ω_Z), does not require the cumbersome feasible boundary constraint projections of [58, 59], and results in significantly fewer function evaluations at the boundary of the search domain than Δ-DOGS(Ω).

Moreover, note that the initialization cost of Δ-DOGS(Ω) is function evaluations at all 2^n vertices of L_s. This initialization cost grows rapidly as dimension n of the

problem increases. The new algorithm, Δ-DOGS(Ω_Z), only requires initial function evaluations at $n + 1$ points.

The following three key modifications to Δ-DOGS(Ω), summarized in Algorithm 3.1, are performed to obtain Δ-DOGS(Ω_Z), as summarized in Algorithm 3.2:

1. All the datapoints in Algorithm 3.2 are restricted to lie on a Cartesian grid, which is occasionally refined as the algorithm proceeds.

2. At each iteration, two different sets of points are considered, S_E and S_U. Function evaluations are available only for the points in S_E; the points in S_U, dubbed *support points*, are used only to regularize the triangulation of the domain, and the uncertainty function which is built upon this triangulation.

3. Two different search functions, $s_c(x)$ and $s_d(x)$, are considered at each iteration. The continuous search function, $s_c(x)$, is designed is minimized over the entire search domain L_s. The discrete search function, $s_d(x)$, is minimized only over the points in S_U.

It is shown in [1, 58, 59] that the irregular behavior of the uncertainty function $e(x)$ close to the boundary of feasibility in many problems causes many additional function evaluations on the boundaries, which can significantly reduce the convergence rate of Δ-DOGS(Ω). The new Δ-DOGS(Ω_Z) algorithm aims to have far fewer datapoints accumulate on the boundary of the search domain. To achieve this, as mentioned above, Algorithm 3.2 makes use of support points, which are used to eliminate function evaluations at points where they are not truly needed. The set of datapoints at each iteration,

Algorithm 3.2 Δ-DOGS(Ω_Z), designed for (grid-based) accelerated minimization of (3.1).

1: Set $k = 0$ and initialize $\ell = 3$. Take the initial set of support points S_U^0 as all 2^n vertices of the feasible domain Ω. Choose at least $n + 1$ points on the initial grid, $n + 1$ of which are affinely independent, put them in S_E^0, and calculate $f(x)$ at each of these $n + 1$ points.
2: Calculate (or, for $k > 0$, update) interpolating functions $p^k(x)$ and $g_\ell^k(x)$ for $f(x)$ and $c_\ell(x)$ over the set of points in S_E^k.
3: Calculate (or, for $k > 0$, update) a Delaunay triangulation Δ^k over all of the points in $S^k = S_U^k \cup S_E^k$, and generate the $e^k(x)$.
4: Find x_k as the minimizer of $s_c^k(x)$ (see Definition (3.6)) in L_s, and take y_k as its quantization onto the grid L_ℓ.
5: Find w_k as the minimizer of $s_d^k(x)$ (see Definition (3.7)) in S_U^k.
6: If the pair (x_k, S^k) is not activated, then take $S_U^{k+1} = S_U^k \cup \{y_k\}$, increment k, and repeat from 2.
7: If $s_d^k(w_k) \leq s_d^k(x_k)$, then take $S_U^{k+1} = S_U^k - \{w_k\}$, $S_E^{k+1} = S_E^k \cup \{w_k\}$, calculate $f(w_k)$ and $c_\ell(w_k)$, and increment k; if $f(w_k) > f_0$ or $c_\ell(y_k) > 0$, repeat from 2, otherwise halt.
8: If $y_k \notin S_E^k$, then take $S_E^{k+1} = S_E^k \cup \{y_k\}$, calculate $f(y_k)$, and increment k; if $f(y_k) > f_0$ or $c_\ell(y_k) > 0$, repeat from 2, otherwise halt.
9: Increment both ℓ and k, and repeat from 2.

S^k, is thus divided into evaluated points S_E^k and support points S_U^k.

Definition 8. *The Cartesian grid of level ℓ for the search domain $L_s = \{x | a \leq x \leq b\}$, denoted as L_ℓ, is:*

$$L_\ell = \left\{ x | x = a + \frac{1}{N}(b-a) \otimes z, \quad z \in \{0, 1, \ldots, N\} \right\}, \text{ where } N = 2^\ell.$$

A quantization of any point $x \in L_s$ on L_ℓ is a point x_q with the minimum distance to x from the L_ℓ grid. See the appendix for a review of essential elements of the Cartesian grid as used in this work.

Algorithm 3.2 aims to have fewer datapoints accumulate on the boundary of the search domain. It is shown in [1, 58, 59] that as the irregular behavior of the

uncertainty function $e(x)$ close to the boundary of feasibility causes many additional function evaluations on the boundaries which can ultimately result in slow convergence.

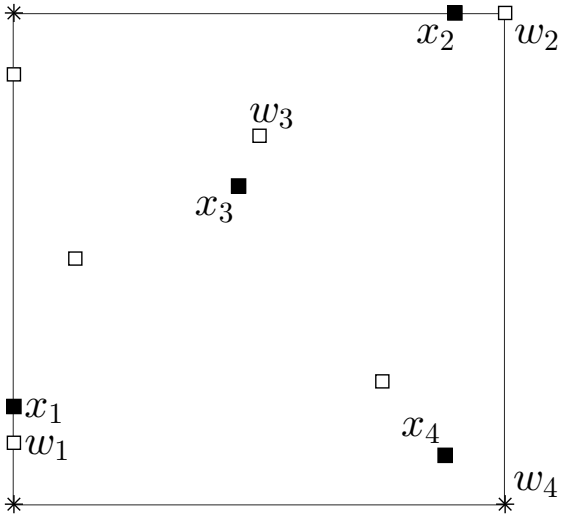
To address this issue Algorithm 3.2, Δ-DOGS(Ω_Z), introduces the notion of "support points", which are points defined and used to eliminate constraint and objective function evaluations on the boundary of the search domain, where these functions are sometimes ill-behaved, while restricting all datapoints to like on a Cartesian grid that is successively refined as convergence is approached.

As a result the datapoints S^k are divided into evaluated points S_E^k and support points S_U^k. That way, Δ-DOGS(Ω_Z) explores the interior of feasible domain more extensively using both continuous and discrete search functions, and the convergence is achieved with fewer function evaluations. It is of note that this issue is more visible when the objective functions themselves have irregular behavior on the boundaries, such as the application of interest in this work.

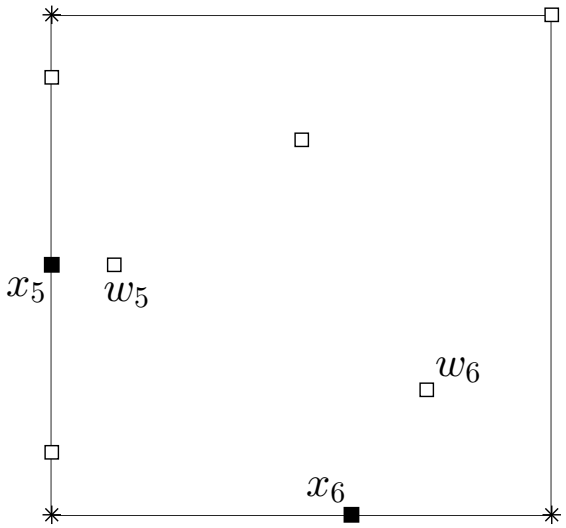
Consider x_k as the minimizer of the continuous search function $s_c^k(x)$ in L_s, y_k as the quantization of x_k onto the grid L_ℓ, and w_k as the minimizer of the discrete search function $s_d^k(x)$ in S_U^k. There are four possible cases at each iteration of Algorithm 3.2, that are corresponding to four of the numbered steps of this algorithm:

(6) The pair (x_k, S^k) is not activated. This is called the *inactivated step*: y_k is simply added to S_U^k, and no function evaluation is performed. (Note that the other three steps below, in contrast, are said to be *activated*.)

(7) The pair (x_k, S^k) is activated and $s_d^k(w_k) < s_d^k(x_k)$. This is called the *replacing step*:



(a) Activated steps. Respective nearest points to $x_{1:4}$ are $w_{1:4}$, on the same set of binding constraints $A_a(x_i) \subseteq A_a(w_i)$.



(b) Inactivated steps. Respective nearest points to $x_{5:6}$ are $w_{5:6}$, not on the same set of binding constraints $A_a(x_i) \nsubseteq A_a(w_i)$.

Figure 3.1: Activated and inactivated steps. Squares: S_E^k, stars: S_U^k.

w_k is removed from S_U^k, added to S_E^k, and $f(w_k)$ calculated.

(8) The pair (x_k, S^k) is activated, $s_d^k(x_k) \leq s_d^k(w_k)$, and $y_k \notin S_E^k$. This is called the *improving step*: the new point y_k is added to S_E^k, and $f(y_k)$ is calculated.

(9) The pair (x_k, S^k) is activated, $s_d^k(x_k) \leq s_d^k(w_k)$, and $y_k \in S_E^k$. This is called the *refinement step*: L_ℓ is refined, and the sets S_E^k and S_U^k are unchanged.

As the algorithm proceeds at a given iteration k of Algorithm 3.2, only one of the above four cases applies, and the corresponding step is taken. Replacing and improving and iterations (in which the replacing and improving steps are taken, respectively) are represented in Fig. 3.1 (note that, in 1D, all iterations are activated).

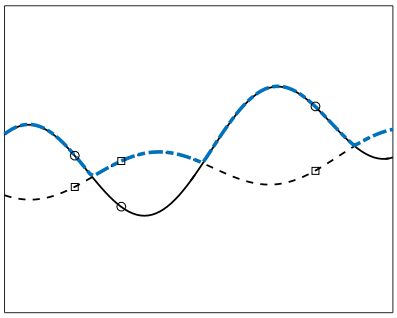
We presented a new algorithm derived from Δ-DOGS(Ω) which, by using Cartesian grids and support points, performs fewer function evaluations at the boundaries of the search domain, and thus has a lower computational cost. The question of whether or not it converges efficiently remains, and will be tackled in the next section.

As the algorithm proceeds at a given iteration k of Algorithm 3.2, only one of the above four cases applies, and the corresponding step is taken. Replacing and improving iterations (in which the replacing and improving steps are taken, respectively) are illustrated in Fig. 3.2; note that, in 1D, all iterations are activated, as the vertices of the domain are included in S_U^k.

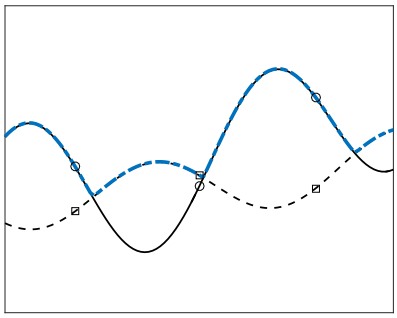
3.3 Convergence analysis of Algorithm 3.2

We analyze the convergence of Algorithm 3.2 under the following assumptions:

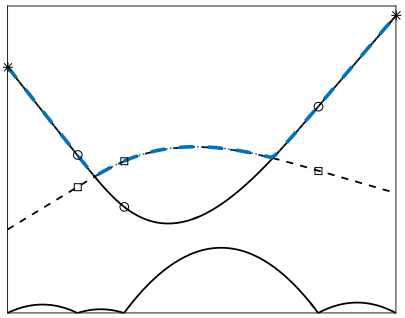
Assumption 4. *The objective function $f(x)$, the constraint functions $c_\ell(x)$, and the interpolating functions $p^k(x)$ and $g_\ell^k(x)$ are all Lipschitz, with Lipschitz constant $\hat{L}$.*



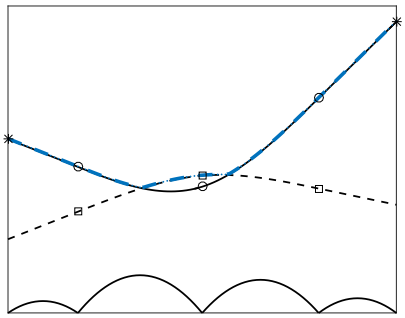
(a) $c_1(x)$ (solid), $c_2(x)$ (dashed), and $\max\{c_1(x), c_2(x)\}$ (blue)



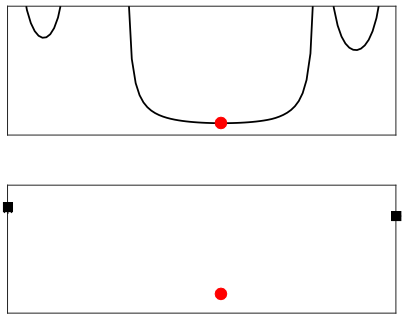
(b) $c_1(x)$ (solid), $c_2(x)$ (dashed), and $\max\{c_1(x), c_2(x)\}$ (blue)



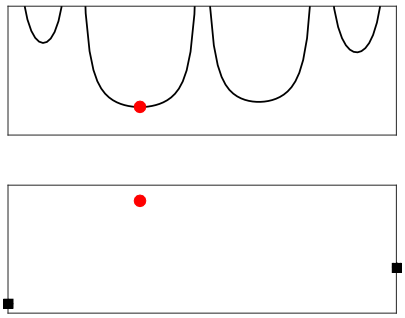
(c) $g_1(x)$ (solid), $g_2(x)$ (dashed), and $\max\{g_1(x), g_2(x)\}$ (blue)



(d) $g_1(x)$ (solid), $g_2(x)$ (dashed), and $\max\{g_1(x), g_2(x)\}$ (blue)



(e) $s_c(x)$ (solid) and $s_d(x)$ (black squares)



(f) $s_c(x)$ (solid) and $s_d(x)$ (black squares)

Figure 3.2: Improving (left) and replacing (right) iterations of Algorithm 3.2. (c,d) Approximations $g_1(x)$ of $c_1(x)$, and $g_2(x)$ of $c_2(x)$. Stars: S_U. Dots, squares: S_E. (e,f) Continuous and discrete search functions. Red dots: global minimizer x_k.

Assumption 5. *The objective function $f(x)$ and constraint functions $c_\ell(x)$ are twice differentiable, and*

$$\nabla^2 f(x) + \hat{K}I > 0, \quad -\nabla^2 f(x) + \hat{K}I > 0, \tag{3.9}$$

$$\nabla^2 c_\ell(x) + \hat{K}I > 0, \quad -\nabla^2 c_\ell(x) + \hat{K}I > 0 \quad \forall 1 \le \ell \le m, \tag{3.10}$$

$$\nabla^2 p^k(x) + \hat{K}I > 0, \quad -\nabla^2 p^k(x) + \hat{K}I > 0 \quad \forall k > 0 \tag{3.11}$$

$$\nabla^2 c_\ell^k(x) + \hat{K}I > 0, \quad -\nabla^2 c_\ell^k(x) + \hat{K}I > 0 \quad \forall 1 \le \ell \le m, \quad \text{and } \forall k > 0. \tag{3.12}$$

Assumption 6. *A point $x \in L_s$ exists such that both $f(x) \le f_0$ and $c_\ell(x) \le 0$. This means that the target value f_0 is achievable within the feasible domain Ω, though a location x that achieves this target value is, at least initially, unknown.*

Lemma 5. *There are infinite number of mesh decreasing steps as Algorithm 3.2 proceeds.*

Proof. Lemma 5 is established in Theorem 1 of [59]. □

Lemma 6. *Consider x^* as a global minimizer of $f(x)$ in Ω. Then, for each step of Algorithm 3.2,*

$$\min\left\{\frac{s_c^k(x^*)}{\hat{K}}, \min_{z \in S_U^k}\{\frac{s_d^k(z)}{\hat{L}}\}\right\} \le 2. \tag{3.13}$$

Proof. Lemma 6 is analogous to Lemma 6 of [59]. □

Another result which is required to establish convergence, which is an extension

of Lemma 2 in [59], is now presented.

Lemma 7. *Consider $J(x)$ and $G_\ell(x)$, for $1 \leq \ell \leq m$, as twice differentiable such that, for some constant K_1, $\nabla^2 J(x) - 2\,K_1 I \leq 0$ and $\nabla^2 G_\ell(x) - 2\,K_1\,I \leq 0$. Also, $J(x) - G_\ell(x)$ and $G_i(x) - G_j(x)$ are Lipschitz functions with Lipschitz constant L_1. Moreover, $x^* \in \Omega$ is a KKT point of the following optimization problem:*

$$\min_{x \in L_s} J(x) \quad \textit{subject to} \quad G_\ell(x) \leq 0 \quad \ell = \{1, \ldots, m\}. \tag{3.14}$$

Then, for each $x \in L_s$ such that $A_a(x^*) \subseteq A_a(x)$, we have:

$$\max\left\{J(x) - J(x^*), \max_{\ell}\{G_\ell(x)\}\right\} \leq K_1 \|x - x^*\|^2 + L_1\,\|x - x^*\|. \tag{3.15}$$

Proof. Since x^* is a KKT point in Ω, it is a stationary point of the following function with respect to x:

$$T(x) = \frac{J(x) + \sum_{\ell=1}^{m} \lambda_\ell G_\ell(x)}{1 + \sum_{\ell=1}^{m} \lambda_\ell}, \tag{3.16}$$

where $\lambda_\ell > 0$ are the Lagrange multipliers. By construction, $\nabla^2 T(x) - 2\,K_1\,I \leq 0$; therefore, according to Lemma 2 in [59], the $T(x)$ can be expressed as:

$$T(x) - T(x^*) \leq K_1 \|x - x^*\|^2. \tag{3.17}$$

On the other hand, define $F(x)$ as follows:

$$F(x) = J(x) - T(x) = \frac{J(x) - \sum_{\ell=1}^{m} \lambda_\ell G_\ell(x)}{1 + \sum_{\ell=1}^{m} \lambda_\ell}$$

It is easy to observe that $F(x)$ is Lipschitz with constant L_1; therefore,

$$|F(x) - F(x^*)| \leq L_1 \|x - x^*\|, \tag{3.18}$$

$$|J(x) - J(x^*)| \leq K_1 \|x - x^*\|^2 + L_1 \|x - x^*\|. \tag{3.19}$$

Similarly, we can show above equation for $G_\ell(x)$ for $1 \leq \ell \leq m$. □

Theorem 4. *Consider k as a mesh refinement step of Algorithm 3.2. Then,*

$$\min_{z \in S_E^k} \Big\{ \max\{f(z) - f_0, \max_{1 \leq \ell \leq m} c_\ell(z)\} \Big\} \leq \max\{3\,\hat{L}\,\delta_k, 6\,\hat{K}\,\delta_k^2\}, \tag{3.20}$$

Recall that z is the quantization x_k on a grid of level ℓ^k, and δ_k is the maximum quantization error on a grid of level ℓ^k (see Definition 8).

Proof. Theorem 4 is analogous to Theorem 1 of [59] which is established similarly using Lemma 7 above (instead of Lemma 2 in [59]). □

We may conclude from the above analysis that, given the assumptions mentioned at the beginning of this section, Algorithm 3.2 converges quadratically as the global minimizer of the optimization problem is approached. Note that Algorithm 3.2 may be applied effectively to many nonconvex optimization problems even when the as-

sumptions of smoothness of the objective and constraint functions do not hold, though the above analysis on the rate of convergence will no longer apply.

3.4 Runge-Kutta scheme derivation

In this section, we develop the constraints which need to be solved in order for us to identify a new mixed implicit/explicit incremental Runge-Kutta (IMEXRK) scheme with low truncation error and third order accuracy. We consider an ordinary differential equation (ODE) with a separable right-hand side:

$$\frac{d\mathbf{u}}{dt} = \mathcal{L}\,\mathbf{u} + \mathcal{N}(\mathbf{u}) \tag{3.21}$$

where $\mathcal{L}$ is a linear operator and $\mathcal{N}$ is a nonlinear operator. This ODE structure is of particular interest since it often arises in the field of Computational Fluid Dynamics (CFD). The incompressible Navier-Stokes Equations (NSE), after appropriate spatial discretization, are an example of the form (3.21). For incompressible NSE, the linear operator accounts for discretization of the diffusive terms, while the nonlinear operator deals with the convective terms. Since the diffusive terms are generally stiff, while the convective terms are usually nonstiff, time discretization for DNS and LES simulations in the past three decades has relied principally on mixed implicit/explicit approaches, in which the integration of the diffusive term is carried out implicitly, while the convective term is marched explicitly.

Our goal is to derive a mixed implicit/explicit (IMEX) Runge-Kutta (RK) algorithm to integrate the equation in (3.21) over time with third order accuracy, while keeping memory storage to a minimum. In this framework, the linear term is treated implicitly, so that the stability of the implicit part is not affected, while the nonlinear term is treated explicitly. Since the linear operator is general stiff, while the nonlinear operator usually is nonstiff, this approach allows us to significantly relax the stability constraints for the time step, with which the simulation is marched.

Recently, [4] showed that is possible to extend the three-step incremental formulation presented in [55] to a four-step scheme, in order to achieve third order accuracy. Such scheme marches the solution $\mathbf{u}_n$ at time t_n over the interval $[t_n,\, t_{n+1}]$ of size Δt as follows

$$\begin{aligned}
\mathbf{u}^{(1)} &= \mathbf{u}_n + \Delta t \left(\alpha_1^I \mathcal{L}\,\mathbf{u}^{(1)} + \beta_1^I \mathcal{L}\,\mathbf{u}_n + \beta_1^E \mathcal{N}(\mathbf{u}_n)\right) \\
\mathbf{u}^{(2)} &= \mathbf{u}^{(1)} + \Delta t \left(\alpha_2^I \mathcal{L}\,\mathbf{u}^{(2)} + \beta_2^I \mathcal{L}\,\mathbf{u}^{(1)} + \beta_2^E \mathcal{N}(\mathbf{u}^{(1)}) + \gamma_2^E \mathcal{N}(\mathbf{u}_n)\right) \\
\mathbf{u}^{(3)} &= \mathbf{u}^{(2)} + \Delta t \left(\alpha_3^I \mathcal{L}\,\mathbf{u}^{(3)} + \beta_3^I \mathcal{L}\,\mathbf{u}^{(2)} + \beta_3^E \mathcal{N}(\mathbf{u}^{(2)}) + \gamma_3^E \mathcal{N}(\mathbf{u}^{(1)})\right) \\
\mathbf{u}_{n+1} &= \mathbf{u}^{(3)} + \Delta t \left(\alpha_4^I \mathcal{L}\,\mathbf{u}_{n+1} + \beta_4^I \mathcal{L}\,\mathbf{u}^{(3)} + \beta_4^E \mathcal{N}(\mathbf{u}^{(3)}) + \gamma_4^E \mathcal{N}(\mathbf{u}^{(2)})\right)
\end{aligned} \tag{3.22}$$

where $\mathbf{u}_{n+1}$ is the solution at time t_{n+1}. Recasting the coefficients in Butcher tableaux

from [61] for explicit and implicit gives

$$
\begin{array}{c|ccccc}
0 & 0 & & & & \\
c_2 & b_1^I & a_{2,2}^I & & & \\
c_3 & b_1^I & b_2^I & a_{3,3}^I & & \\
c_4 & b_1^I & b_2^I & b_3^I & a_{4,4}^I & \\
1 & b_1^I & b_2^I & b_3^I & b_4^I & b_5^I \\
\hline
 & b_1^I & b_2^I & b_3^I & b_4^I & b_5^I
\end{array}
\qquad
\begin{array}{c|ccccc}
0 & 0 & & & & \\
c_2 & a_{2,1}^E & 0 & & & \\
c_3 & b_1^E & a_{3,2}^E & 0 & & \\
c_4 & b_1^E & b_2^I & a_{4,3}^E & 0 & \\
1 & b_1^E & b_2^E & b_3^E & b_4^E & 0 \\
\hline
 & b_1^E & b_2^E & b_3^E & b_4^E & 0
\end{array}
\tag{3.23}
$$

This alternative formulation allows us to easily impose the nine constraints needed in order for the scheme to achieve full third order accuracy during the integration of (3.21). Such constraints are listed below:

$$
\begin{aligned}
\tau_1^{(1)I} &= \sum_{i=1}^{s} b_i^I - 1 & \tau_1^{(1)E} &= \sum_{i=1}^{s} b_i^E - 1 \\
\tau_1^{(2)I} &= \sum_{i=1}^{s} b_i^I \, c_i - \frac{1}{2} & \tau_1^{(2)E} &= \sum_{i=1}^{s} b_i^E \, c_i - \frac{1}{2} \\
\tau_1^{(3)E} &= \frac{1}{2} \sum_{i=1}^{s} b_i^E \, c_i^2 - \frac{1}{6} & & \\
\tau_2^{(3)II} &= \sum_{i,j=1}^{s} b_i^I \, a_{i,j}^I \, c_i - \frac{1}{6} & \tau_2^{(3)IE} &= \sum_{i,j=1}^{s} b_i^I \, a_{i,j}^E \, c_j - \frac{1}{6} \\
\tau_2^{(3)EI} &= \sum_{i,j=1}^{s} b_i^E \, a_{i,j}^I \, c_i - \frac{1}{6} & \tau_2^{(3)EE} &= \sum_{i,j=1}^{s} b_i^E \, a_{i,j}^E \, c_j - \frac{1}{6}
\end{aligned}
\tag{3.24}
$$

Furthermore, we impose stage-order one constraints at each stage of the Butcher tableaux in (3.23), i.e. $\sum_{j=1}^{5} a_{i,\,j}^I = \sum_{j=1}^{5} a_{i,\,j}^E = c_i$. This leaves three free design parameters for

optimization. All in all, after third order accuracy is imposed, we should minimize the fourth order truncation error. This is accomplished in [4, 62, 63] by minimizing the norm of the residuals of the fourth order accuracy constraints, denoted as $A^{(4)}$.

For the ODE in (3.21), such constraints are

$$\begin{aligned}
\tau_2^{(4)EI} &= \sum_{i,j=1}^{s} b_i^E\, c_i\, a_{i,j}^I\, c_j - \frac{3}{24} & \tau_2^{(4)EE} &= \sum_{i,j=1}^{s} b_i^E\, c_i\, a_{i,j}^E\, c_j - \frac{3}{24} \\
\tau_3^{(4)IE} &= \frac{1}{2}\sum_{i,j=1}^{s} b_i^I\, a_{i,j}^E\, c_j^2 - \frac{1}{24} & \tau_3^{(4)EE} &= \frac{1}{2}\sum_{i,j=1}^{s} b_i^E\, a_{i,j}^E\, c_j^2 - \frac{1}{24} \\
\tau_4^{(4)III} &= \sum_{i,j,k=1}^{s} b_i^I\, a_{i,j}^I\, a_{j,k}^I\, c_k - \frac{1}{24} & \tau_4^{(4)IIE} &= \sum_{i,j,k=1}^{s} b_i^I\, a_{i,j}^I\, a_{j,k}^E\, c_k - \frac{1}{24} \\
\tau_4^{(4)IEI} &= \sum_{i,j,k=1}^{s} b_i^I\, a_{i,j}^E\, a_{j,k}^I\, c_k - \frac{1}{24} & \tau_4^{(4)IEE} &= \sum_{i,j,k=1}^{s} b_i^I\, a_{i,j}^E\, a_{j,k}^E\, c_k - \frac{1}{24} \\
\tau_4^{(4)EII} &= \sum_{i,j,k=1}^{s} b_i^E\, a_{i,j}^I\, a_{j,k}^I\, c_k - \frac{1}{24} & \tau_4^{(4)EIE} &= \sum_{i,j,k=1}^{s} b_i^E\, a_{i,j}^I\, a_{j,k}^E\, c_k - \frac{1}{24} \\
\tau_4^{(4)EEI} &= \sum_{i,j,k=1}^{s} b_i^E\, a_{i,j}^E\, a_{j,k}^I\, c_k - \frac{1}{24} & \tau_4^{(4)EEE} &= \sum_{i,j,k=1}^{s} b_i^E\, a_{i,j}^E\, a_{j,k}^E\, c_k - \frac{1}{24}.
\end{aligned} \tag{3.25}$$

In practice, the IMEXRK scheme should have an implicit component, which is L-stable[2]. This guarantees proper damping of the largest eigenvalues of $\mathcal{L}$. Based on linear stability analysis [64, 65], the stability function for the implicit component of (3.23) can be defined as

$$\sigma^I(z^I) = \frac{\sum_{i=0}^{3} p_i^I [z^I]^i + p_4^I [z^I]^4}{\sum_{i=1}^{3} q_i^I [z^I]^i + q_4^I [z^I]^4}, \tag{3.26}$$

[2] A time marching scheme is said to be L-stable if its stability region contains the entire left-half plane (LHP), and $\sigma(\infty) = \lim_{z\to\infty} \sigma(z) = 0$.

where p_4^I, q_4^I, p_i^I, and q_i^I are functions of the Butcher coefficients $a_{i,j}^I$, b_i^I, and c_i, while $z^I = \lambda\Delta t$, where λ is an eigenvalue of $\mathcal{L}$. L-stability is achieved when the stability region $|\sigma^I(z^I)| \leq 1$ covers the entire left-half plane and $\sigma^I(z^I)$ goes to zero as z^I approaches infinity. Algebraically, this is equivalent to imposing $p_4^I = 0$, provided that q_4^I does not vanish (exists and does not approach infinity) so that $|p_4^I/q_4^I|$ approaches zero. If L-stability cannot be achieved, a strong A-stability should be sought. This is equivalent to satisfying

$$\sigma_\infty = \lim_{z^I \to \infty} |\sigma(z^I)| = |p_4^I/q_4^I| < 1. \tag{3.27}$$

Another important property for IMEXRK schemes is the extension of the stability region for the explicit component. For fluid dynamics application, the extension along the imaginary axis is of primary interest, since it directly relates to the Courant-Friedrichs-Lewy (CFL) condition. For the scheme in (3.23), the stability function for the explicit component $\sigma^E(z^E)$ reads

$$\begin{aligned} \sigma^E(z^E) = 1 + \sum_i b_i^E\, z^E + \sum_i b_i^E\, c_i [z^E]^2 + \\ + \sum_{i,j} b_i^E\, a_{i,j}^E\, c_j [z^E]^3 + \sum_{i,j,k} b_i^E\, a_{i,j}^E\, a_{j,k}^E\, c_k [z^E]^4, \end{aligned} \tag{3.28}$$

where $z^E = \mu\Delta t$, where μ is the eigenvalue of the linearization of the operator $\mathcal{N}$ at a given instant t_n. After imposing third order accuracy constraints $\tau_1^{(1)E} = \tau_1^{(2)E} = \tau_2^{(3)EE} = 0$, the expression for $\sigma^E(z^E)$ now reads

$$\sigma^E(z^E) = 1 + z^E + [z^E]^2/2 + [z^E]^3/6 + \delta\,[z^E]^4, \tag{3.29}$$

where $\delta = \sum_{i,j,k} b_i^E \, a_{i,j}^E \, a_{j,k}^E \, c_k$. In [4] $\delta = 1/24$ was found to give the maximum extension of the stability region $|\sigma^E| \leq 1$ along the imaginary value, achieving the value $2\sqrt{2}$. We also found in [4] that a value $\delta > 1/24$ produces a stability region, which does not include the entire imaginary axis between the origin of the complex plane and the farthest intersection point between the imaginary axis and the stability region. For this reason, only those schemes with a set of coefficients for which

$$\delta \leq \frac{1}{24} \tag{3.30}$$

will be considered during the optimization stage.

Most importantly, these properties must be achieved while ensuring that the coefficients are all real-valued and reasonably small.This is a practical aspect that simplifies the implementation and reduces the impact of algebraic error during the time integration. Another condition that is generally imposed is that all the c_i coefficients must be in the interval [0, 1]. This ensures that all the function evaluations needed for the time integration over $[t_n, \, t_{n+1}]$ are performed using points within $[t_n, \, t_{n+1}]$.

3.5 Formulation of the Optimization Problem

In this section, using the analysis we developed in section 3.4, we design the optimization problem.

Let us consider $\mathbf{c} = [c_2, c_3, c_4]$, the free parameters for our optimization algo-

rithm. Based on the considerations in Section 3.4, a box domain $[0, 1]^3$ was chosen to perform optimization over the parameter space.

At each iteration of the optimization algorithm, the nine constraints needing to achieve third order accuracy are imposed as follows. First, equations $\tau_1^{(1,2,3)E} = 0$, and $\tau_1^{(1,2)I} = 0$ are solved together with stage-order one condition. Notice that once the $\mathbf{c}$ coefficients have numerical values, these equations are linear in the b_i^E coefficients and are easily solvable. Provided the solution $\mathbf{c}$ at the current iteration does not cause the linear system to become singular, the coefficients $b_{1,2,3}^E$ and $b_{1,2}^I$ are determined at this stage as a function of the remaining coefficients. Afterward, the resulting expressions are replaced into $\tau_2^{(3)EE} = 0$. The arising second order equation is then solved for b_4^E. Provided the radicand Δ_E of $\tau_2^{(3)EE} = 0$ is non-negative, two choices for b_4^E are obtained. Replacing the values for b_4^E back into $\tau_1^{(1,2,3)E} = 0$ allows to completely determine the set of coefficients b_i^E. Such values and the expressions for $b_{1,2}^I$ are then replaced into $\tau_2^{(3)EI} = 0$ and $\tau_2^{(3)IE} = 0$. These two equations, together with stage-order one condition for the implicit part, are linear in the b_i^I parameters and are used to determine $b_{3,4}^I$. Substitution of the resulting expressions into $\tau_2^{(3)II} = 0$ gives a second order equation in b_5^I. Notice that since two values of b_4^E have been found, we have two quadratic equations to be solved for b_5^I, this means that, provided the radicands $\Delta_{I(1,2)}$ are positive, we obtain four solutions for b_5^I. Replacing the values obtained in the previous expressions allows to completely determine the coefficient set.

This subroutine can be implemented within the optimization algorithm. However, such an automatized approach breaks down whenever the initial linear system is

singular. The singularity can be avoided by two constraints: a) by tightening the bounds of the search domain to avoid the boundary points, 0 and 1. This is achieved by restricting the search domain to the interval $[tol_c,\ 1 - tol_c]$[3]. b) by imposing additional constraints in the optimization, i.e. $|c_i - c_j| \geq tol_c$, for $i \neq j$ to avoid equality between c_i coefficients.

Notice that this approach prevents from exploring among those schemes (regions) associated to these particular solutions and an *ad hoc* optimization problem should be set up for each of these choices. Furthermore, it is necessary to impose that the solution found is acceptable, i.e. the the coefficients are all real-valued. This is achieved by enforcing the extra constraints $\Delta_E,\ \Delta_{I(1,2)} \geq 0$ during the optimization.

These constraints have a nonlinear behaviour respect to the c_i parameters and become discontinuous in the parameter space of the solutions. In order to reduce the discontinuity and bound the objective function and the constraint functions the hyperbolic tangent function is used as a saturation function.

The optimization problem can be cast as follows:

$$\begin{aligned} \min_{\mathbf{c}} \quad & A^{(4)} \\ \text{subject to} \quad & \sigma_\infty = \frac{p_4^I}{q_4^I} = 0 \\ & 1/24 - \delta = 0 \\ & \Delta_E \geq 0 \\ & \Delta_{I(1,2)} \geq 0 \\ & |c_i - c_j| - tol_c \geq 0, \text{ for } i \neq j \\ & tol_c \leq c_2, c_3, c_4 \leq 1 - tol_c \end{aligned} \tag{3.31}$$

There are some thresholds for each constraint and we can write the problem (3.31) in a way that would in form of adaptive-K algorithm.

With the powerful new derivative-free optimization method we developed, we were able to design a new IMEX Runge Kutta scheme which is third-order accurate. This Runge-Kutta scheme is useful for solving the stiff ODEs resulting from high performance computing works, like turbulence simulations. We will now present numerical results we get from the application of these new schemes.

3.6 Results

In this work, we first developed an efficient derivative-free scheme that handles problems with multiple nonconvex constraints. We compared the newly developed op-

timization method with the method developed in [58] on the application-based problem to design a new IMEX Runge Kutta scheme, as explained in §3.4.

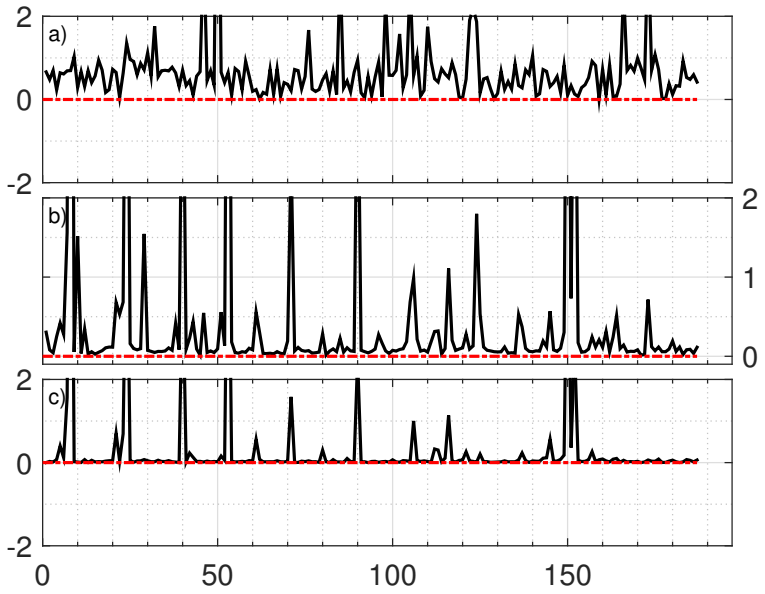
In this section, the discovered IMEXRK scheme is compared to the most commonly available scheme that is being used in the Turbulent community. Finally, the properties of our new scheme are illustrated on a representative example: the Burger equation.

3.6.1 Comparison between the basic and modified optimization methods

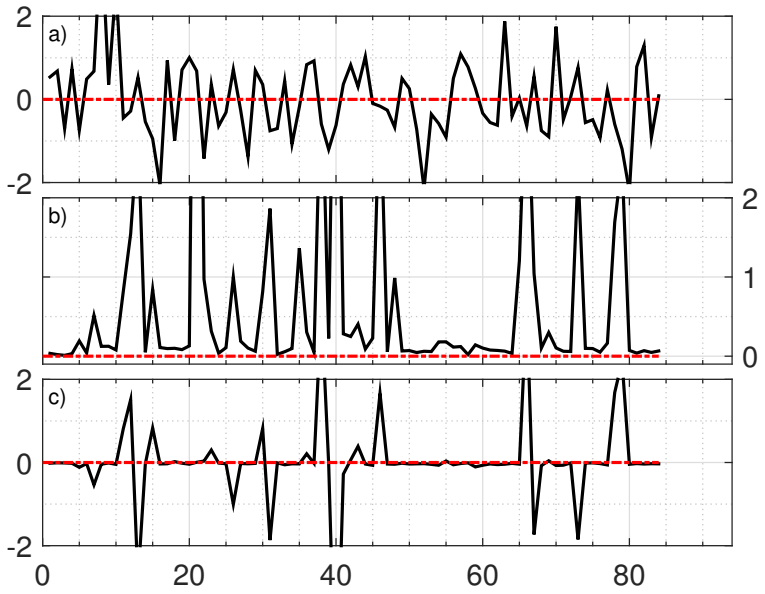
In this part, we compare the performance of Algorithm 3.2 with the original Δ-DOGS(Ω) algorithm as summarized in [58] on optimization problem (3.31). Both these schemes are applied on the (3.31) problem, which is a relatively computationally expensive, nonlinear optimization problem with non-analytical constraint and objective functions that is well suited for the use of Δ-DOGS(Ω) algorithms.

Using the Δ-DOGS(Ω) algorithms, several new mixed Implicit/Explicit Runge Kutta schemes have been discovered which are appropriate for the simulation of incompressible turbulence flow.

Note that the constrained optimization problem (3.31) is nonlinear and most of the commercial off-the-shelf (COTS) available optimization method fail to solve this problem and find an acceptable solution. Moreover, the brute-force algorithms cost more than hundred thousands of function evaluations.



(a) Δ-DOGS(Ω)



(b) Δ-DOGS(Ω_Z)

Figure 3.3: Results from the basic vs modified algorithm. a) L-Stability σ_∞. b) Truncation error $A^{(4)}$. c) Parameter $(1/24 - \delta)$.

Fig. 3.3 shows this improvement and compare the results for both optimization algorithms. The top figure shows the search function $s(x)$. This search function in fact represents the maximum violation of constraints and the difference between the target value f_0 and the best available point.

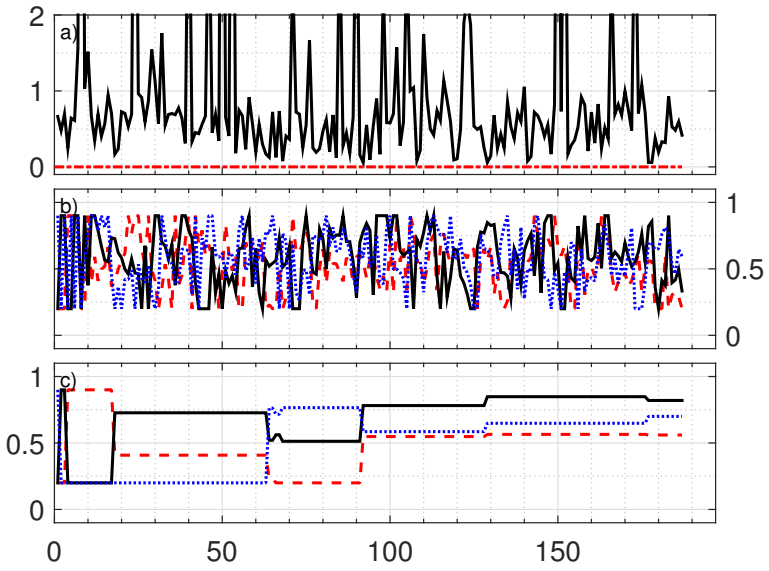
In the situations that there is a rapid change in the objective and constraint values close to the boundary it is beneficial to quantize the points to a lattice instead of using the constraint projection method [37]. We stop the algorithm when we have a repeated point on the lattice and all the constraints are satisfied in that point.

Fig. 3.4 shows the trajectory of the design parameters using both optimization algorithms. In both plots the top plot shows the maximum violation or the search function trajectory and the red dashed line the target value that the algorithm is seeking.

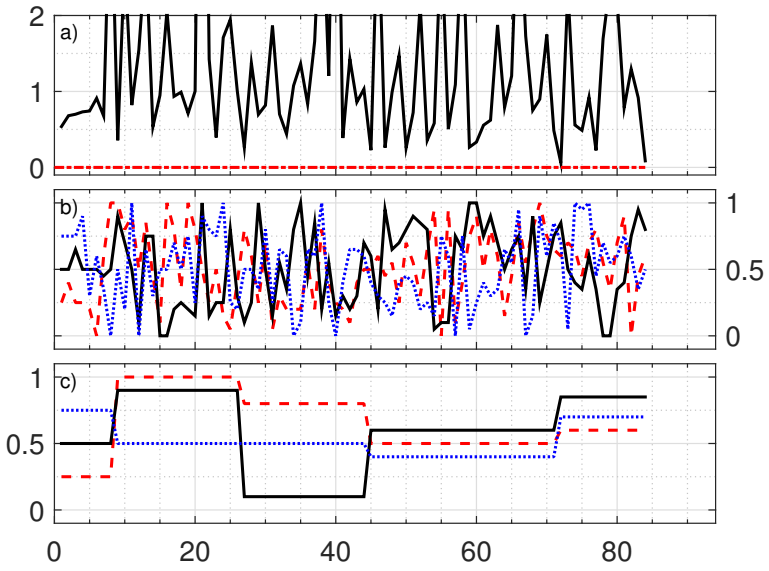
It is shown in Fig. 3.4 that to find an acceptable solution, the Δ-DOGS(Ω_Z) uses half the computational cost of the Δ-DOGS(Ω). Most of the extra function evolutions are performed close to the boundary of the solutions which in advance we knew that is not the case. Δ-DOGS(Ω_Z) generates the support points, defined to regularize the irregular behaviour of the objective function on the boundaries.

The target value for both optimization algorithms for the objective function $A^{(4)} \leq 0.08$ and for the constraint violation the limits of $-0.05 \leq \sigma_\infty \leq 0.05$ and $-0.0001 \leq \delta - 1/24 \leq 0$, $\Delta_E \geq 0.001$ and $\Delta_{I(1,2)} \geq 0.001$, and $|c_i - c_j| - 0.1 \geq 0$.

Table 3.1 illustrates optimized coefficients and error measures for the IMEXR-KiCB3(4s) scheme reported in [4], and the two new schemes developed in this work. The IMEXRKiACBB3a(4s) scheme was found using the Δ-DOGS(Ω) algorithm in



(a) Δ-DOGS(Ω)



(b) Δ-DOGS(Ω_Z)

Figure 3.4: Results from the basic vs modified algorithm. a) Search function value series. b) Trajectory of the c_i points. c) Trajectory of the solution.

Table 3.1: Optimized coefficients and error measures for the IMEXRKiCB3(4s) scheme reported in [4], and the two new schemes developed in this work.

IMEXRK	c_2	c_3	c_4	$A^{(4)}$	σ_∞	$1/24 - \delta$
IMEXRKiCB3(4s) [4]	0.560	0.800	0.700	0.059	0.032	−0.003
IMEXRKiACBB3a(4s)	0.5601	0.820	0.700	0.058	0.024	−0.003
IMEXRKiACBB3b(4s)	0.570	0.895	0.725	0.063	−0.008	−0.00007

187 iterations, and the IMEXRKiACBB3a(4s) scheme was found using the new Δ-DOGS(Ω_Z) algorithm, taking $N = 160$, in just 88 iterations.

Thus, we can conclude that our new algorithm shows improved performance and has a lower computational cost when compared with the original algorithm it is derived from.

3.6.2 Evaluation of the scheme on the 1D Burgers Equation

In order to verify the full third-order accuracy for the innovative scheme we developed, it was tested on the time integration of the one-dimensional Burgers Equation (BE):

$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) + \nu\frac{\partial^2 u}{\partial x^2} \tag{3.32}$$

The spatial domain considered is $x \in [0,\, 400\,m]$ with 1024 equally-spaced grid points and pseudo-spectral approach is adopted for the discretization of all spatial derivatives, with 2/3-dealiasing rule for the computation of the convective term. Furthermore, the integration of the diffusive component is carried out implicitly, assuming $\nu = 1$, while the convective term is integrated explicitly. The equation in (3.32) is integrated over a

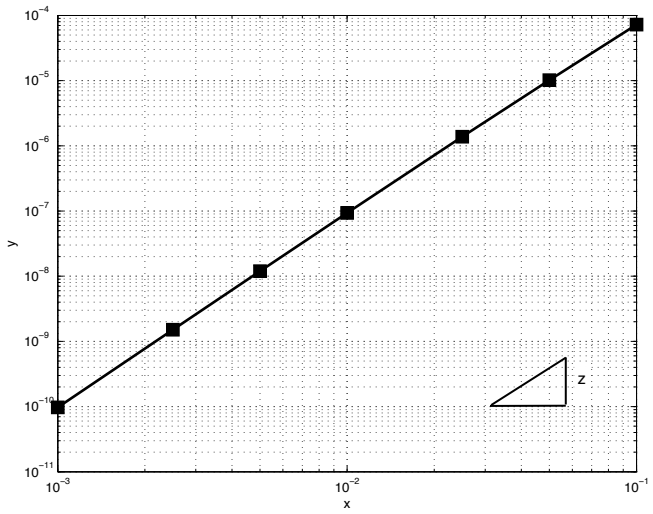
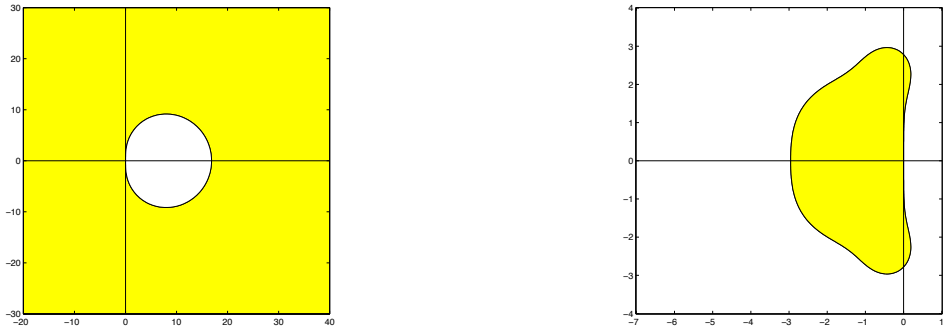


Figure 3.5: Relative error as a function of the step size for the scheme derived in this chapter when applied to the integration of (3.32).

time interval of 10 seconds from the initial condition

$$u(x,\, 0) = e^{-(x-200)^2} \tag{3.33}$$

Error convergence is determined by comparing the solutions obtained with different time steps, ranging from 10^{-3} to 10^{-1}, with the reference solution obtained using the fourth-order IMEXRK scheme ARK4(3)6L[2]SA from [63] with a constant time step $\Delta t = 10^{-5}$. Results are presented in Figure 3.5 and show good agreement with theoretical expectation.



(a) Implicit component. (b) Explicit component.

Figure 3.6: Stability regions for the scheme derived from IMEXRKiACBB3b(4s). Note that the difference in the stability region for IMEXRKiACBB3a(4s) and IMEXRKACBB3b(4s) visually is indistinguishable.

3.6.3 Summary of the comparison performance for the new IMEXRK scheme

The coefficients obtained are listed in Table 3.2 in both Butcher tableaux and incremental form, while stability and accuracy properties are shown in Table 3.1. Stability regions for the implicit and explicit part are shown in Figure 3.6.

Our new scheme shows enhanced computational performance when compared to the original algorithm it has been derived from. Moreover, it has been successfully applied to the one-dimensional Burgers equation, giving results which are in good agreement with the theoretical expectation. These numerical results allow us to conclude that our innovative scheme performs well and is an improvement of existing solutions.

Table 3.2: Optimal parameters for the new IMEXRK scheme derived in this chapter.

Butcher coefficients	
Parameter	Value
b_1^I	35965327958/140127563663
b_2^I	353083323889/1136747549899
b_3^I	−360566052281/1494955198897
b_4^I	756596001291/1512335944289
b_5^I	189462239225/1091147436423
$a_{2,2}^I$	343038331393/1130875731271
$a_{3,3}^I$	288176579239/1140253497719
$a_{4,4}^I$	253330171251/677500478386
b_1^E	829462852521/3426433096921
b_2^E	103183819801/448156083531
b_3^E	9976429300/709197748683
b_4^E	113091689455/220187950967
$a_{2,1}^I$	14/25
$a_{3,2}^I$	777974228744/1346157007247
$a_{4,3}^I$	251277807242/1103637129625
c_2	14/25
c_3	41/50
c_4	7/10

Incremental-form coefficients	
Parameter	Value
α_1^I	343038331393/1130875731271
β_1^I	35965327958/140127563663
α_2^I	288176579239/1140253497719
β_2^I	19632212512/2700543775099
α_3^I	253330171251/677500478386
β_3^I	−173747147147/351772688865
α_4^I	189462239225/1091147436423
β_4^I	91958533623/727726057489
β_1^E	14/25
γ_1^E	0
β_2^E	777974228744/1346157007247
γ_2^E	−251352885992/790610919619
β_3^E	251277807242/1103637129625
γ_3^E	−383714262797/1103637129625
β_4^E	113091689455/220187950967
γ_4^E	−403360439203/1888264787188

3.7 Conclusions

This work introduced and applied a powerful new variant, dubbed Δ-DOGS(Ω_Z), of our lab's Delaunay-based Derivative-free Optimization via Global Surrogates family of algorithms to the practical problem of identifying a new, low-storage, high-accuracy, Implicit/Explicit Runge-Kutta (IMEXRK) time integration scheme for high performance computing (HPC) applications, like the simulation of turbulence. The optimization scheme developed and used in this work, which is provably globally convergent under the appropriate assumptions, combined the essential ideas of (a) our Δ-DOGS(Ω) algorithm, which is designed to efficiently optimize a nonconvex objective function $f(x)$ within a nonconvex feasible domain Ω described by a number of constraint functions $c_\ell(x)$, with (b) our Δ-DOGS(Z) algorithm, which aims to reduce the number of function evaluations on the boundary of the feasible domain that would otherwise be called for via the restriction that all function evaluations lie on a Cartesian grid, which is subsequently refined as the iterations proceed, over the rectangular search domain L_s considered. The identification of the optimal parameters of IMEXRK schemes involved (1) a complicated set of nonlinear constraints, which are imposed in order to achieve the desired order of accuracy in addition to a handful of important stability properties, which leads to a highly nonconvex, disconnected feasible domain, and (2) a highly nonconvex objective function, which represents a compromise between a few different measures characterizing the leading-order error and potential stability shortcomings of the resulting scheme. This structure makes the computation of new IMEXRK schemes a challenging

and well-suited practical test problem for global optimization algorithms to solve. In this work, the new optimization algorithm developed, Δ-DOGS(Ω_Z), introduced the notion of "support points", which are points defined and used to eliminate constraint and objective function evaluations on the boundary of the search domain, where these functions are sometimes ill-behaved, while restricting all datapoints to like on a Cartesian grid that is successively refined as convergence is approached. For validation, the convergence of Δ-DOGS(Ω_Z) and Δ-DOGS(Ω) are compared on a challenging problem of optimizing a low-storage IMEXRK formulation. Results indicate a notably accelerated convergence rate using Δ-DOGS(Ω_Z). In the end, a low-storage third-order accurate IMEXRK algorithm for the time integration of stiff ODEs was identified which exhibited remarkably good stability and accuracy properties as compared with existing IMEXRK schemes.

Using more intelligent function evaluations in the control domain can cause a significant reduction in time and computation. The number of function evaluations is significantly reduced, if compared to *brute-force* approach.

As future work, this optimization problem can be applied to the solution of other optimization problems with complicated constraint functions. Also, the new IMEXRK scheme is applied to the Turbulence flow simulation.

Acknowledgements

Chapter 3 is currently being prepared for submission for publication. S. R. Alimo, D. Cavaglieri, P. Beyhaghi, T.R. Bewley, "Discovery of an IMEXRK time inte-

gration scheme via Delaunay-based derivative-free global optimization ", *Journal of Global Optimization*. The dissertation author was the primary researcher and author of this material.

Chapter 4

Optimization combining derivative-free global exploration with derivative-based local refinement

Everything in the universe is within you. Ask all from yourself.—Rumi

4.1 Introduction

Consider the optimization of a nonconvex, expensive-to-compute function $f(x)$ with bound constraints,

$$\text{minimize } f(x) \text{ with } x \in B = \{x | a \leq x \leq b\}, \tag{4.1}$$

where a and b are two vectors in $\mathbb{R}^n$ such that $a < b$. Solving an optimization problem of the form (4.1) is difficult and, for general functions, convergence can only be guaranteed if the function evaluation set becomes dense everywhere in B as $k \to \infty$ [66]. In this paper, we will restrict our attention to problems in which $f(x)$ is smooth (twice differentiable), and for which the optimization problem considered has a target value f_0; that is, we seek a point $x \in B$ that is a local minimum such that $f(x) \leq f_0$.

There are two classes of optimization algorithms available to solve (4.1): *derivative-based* methods, which use gradient information to accelerate the search of a local minimum of the objective function, and *derivative-free* methods, which do not use gradient information, but may often be developed in a manner which, under the appropriate assumptions, assures convergence to a global solution to (4.1) [67, 68, 69].

Derivative-based methods are designed to handle a large number of design parameters, and generally require far fewer function evaluations for local convergence. There are two main approaches for determining the update made at each step of a derivative-based search: those based on *trust regions*, and those based on *line searches*.

Trust region methods define, at each iteration, a region in the vicinity of the current point, x_k, within which a model that approximates the objective function is generated and used to calculate the next point, which is restricted to lie within the trust region. Line search methods, in contrast, determine the step length in a chosen search direction via a number of additional function evaluations, in order to identify a point with a reduced function value (see [70, 71]), often coordinated by an *Armijo condition*, which seeks to ensure that the next iterate reduces the function sufficiently relative to the

directional derivative of $f(x)$ at x_k in the search direction, or a *Wolfe condition*, which seeks to enforce conditions on ∇f as well as the Armijo condition in order to guarantee that a BFGS update can be safely applied.

Derivative-free methods can, under the appropriate assumptions, guarantee convergence to a global optimum, but are generally quite inefficient computationally, particularly at local refinement, requiring many more function evaluations than derivative-based methods. Response surface methods (RSMs) are the most efficient and widely-used globally-convergent derivative-free optimization methods available today. RSMs iteratively minimize a search function using an interpolant (or, a regressor) of existing data points, known as the "surrogate", and a model of the uncertainty of this surrogate, which generally goes to zero at the function evaluations themselves. Efficient global optimization (EGO) [72], optimization by radial basis function interpolation in trust-regions (ORBIT) [68], the Surrogate-Management-Framework (SMF) [29], and Delaunay-based derivative-free optimization via global surrogates (Δ-DOGS) [1, 37, 58], are modern examples of RSMs.

The derivative-free scheme upon which the present work is based is Δ-DOGS, which is a generalizable family of computationally-efficient optimization algorithms developed by our group for low-dimensional optimization problems in which the objective function is both nonconvex and expensive to evaluate. There are already a handful of schemes in this family, including schemes designed specifically for simple bound constraints [59], linear constraints [37, 73], and nonconvex constraints [58, 60].

This paper proposes the hybridization of (derivative-free) algorithms in the Δ-

DOGS family with a local derivative-based optimization approach in order to significantly accelerate the process of local refinement (for a related discussion, see [69]). The proposed hybrid algorithm inherits the property of global convergence (under the appropriate assumptions) of the particular Δ-DOGS algorithm upon which it is based. In our numerical experiments, the algorithm is found to efficiently handle nonconvex functions with many local minima, and to scale better with dimension than purely derivative-free global optimization approaches.

The structure of this paper is as follows: Section 4.2 briefly reviews the essential ideas of the Δ-DOGS(Z) algorithm, which accelerates a Δ-DOGS search by coordinating it with a Cartesian grid over parameter space that is successively refined as convergence is approached. Section 4.3 explains the new hybrid optimization scheme, which combines Δ-DOGS(Z) with a derivative-based optimization algorithm, leveraging a trust region approach, for local refinement. Section 4.4 analyzes the new hybrid algorithm's convergence properties, and describes the technical conditions needed to guarantee its convergence to a global or local minimizer. In Section 4.5, the hybrid algorithm is applied to benchmark optimization problems to illustrate its behavior. Conclusions are presented in Section 4.6.

4.2 A brief review of Δ-DOGS(Z)

We now review the essential elements of Δ-DOGS(Z) [59, 73]. Note that this paper focuses on variants of these algorithms that leverage target values of f_0; other

variants of these algorithms are discussed in [1, 37, 73, 74].

Δ-DOGS(Z) is an iterative method that sequentially estimates the location in the feasible domain B with the highest probability, given the current surrogate model, of having a function value less than or equal to f_0 (that is, which maximizes a probability measure for finding such a function value). The approach is akin to the expected improvement [75, 76] and Bayesian optimization algorithms [77].

Definition 9. *Consider $S = \{x_1, x_2, \ldots, x_N\}$ as a set of datapoints in the feasible domain B. The continuous search function $s(x)$ is defined as follows:*

$$s(x) = \begin{cases} \frac{p(x)-f_0}{e(x)} & \text{if} \quad p(x) \geq f_0, \\ p(x) - f_0 & \text{otherwise}, \end{cases} \tag{4.2}$$

where $p(x)$ is an interpolating function such that $p(x_i) = f(x_i), \forall i \in \{1, 2, \ldots, N\}$, and $e(x)$ is an uncertainty function built on the framework of a Delaunay triangulation of existing datapoings; key properties of $e(x)$, as discussed further in [37], include it being piecewise quadratic with a constant Hessian, $e(x) \geq 0 \ \forall x \in B$, and $e(x_i) = 0 \ \forall i \in \{1, 2, \ldots, N\}$.

Definition 10. *The Cartesian grid of level ℓ for the feasible domain $B = \{x | a \leq x \leq b\}$, denoted B_ℓ, is defined such that*

$$B_\ell = \left\{ x | x = a + \frac{1}{N}(b-a) \otimes z, \quad z \in \{0, 1, \ldots, N\} \right\}$$

where $N = 2^\ell$. A quantizer of a point $x \in B$ onto B_ℓ is a point x_q on the B_ℓ grid with minimum distance to x; note that the quantizer so defined is not necessarily unique. The maximum discretization error is defined as

$$\delta_\ell = \max_{x \in B} \|x - x_q\|. \tag{4.3}$$

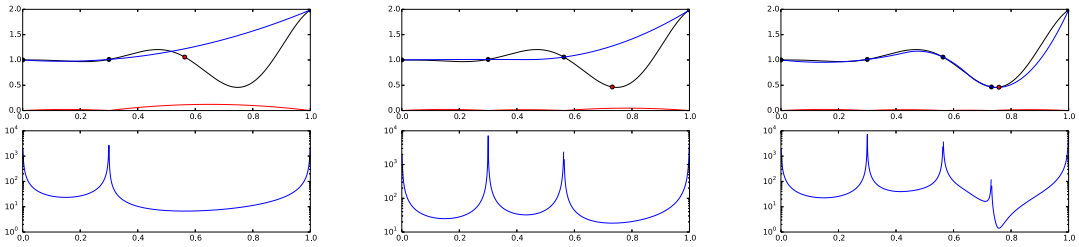
Illustration of the above concepts can be found in Figure 2 of [59]. The Cartesian grid defined above has a specific property that is useful in this analysis: if any constraints on B are binding at x, these constraints are also binding at x_q.

Given the above concepts, Algorithm 4.1 presents a strawman form of the Δ-DOGS(Z) algorithm; more details may be found in [59, 73]. Illustration of an iteration

Algorithm 4.1 Strawman of Δ-DOGS(Z), designed for minimizing $f(x) \in B$ leveraging the target value f_0.

0. Initialize $k = 0$, ℓ, and the initial set of datapoints S_0, and calculate $f(x_i)$ for all $x_i \in S_0$.
1. Calculate or update the interpolating function $p_k(x)$ and the uncertainty function $e_k(x)$ for the points in S_k.
2. Minimize the search function (10) in B to obtain $\hat{x}_k$ as a point with high probability of obtaining the target value.
3. Determine y_k as the quantization of $\hat{x}_k$ on B_{ℓ_k}.
4. If $y_k \notin S_k$, $S_{k+1} = S_k \cup y_k$, and calculate $f(x_k)$; otherwise, refine the mesh by incrementing ℓ_k.
5. Repeat steps 1-4 until a point x is found with $f(x) \leq f_0$.

of Algorithm 4.1 is shown in Figure 4.1 for 3 subsequent iterations.



(a) iteration 1. (b) iteration 2. (c) iteration 3.

Figure 4.1: Essential elements of Algorithm 4.1. Top: truth function $f(x)$ (black), interpolating surrogate function $p(x)$ (blue), synthetic model of the uncertainty $e(x)$ (red), previous (black dots) and new datapoints (red dots). Bottom: search function $s(x)$ (4.2).

Remark 10. *At each iteration, Algorithm 4.1 either adds a new point to the feasible domain, or refines the mesh.*

There are two possible termination scenarios for Algorithm 4.1: either it finds a point x with a function value $f(x) \leq f_0$, or it conducts an infinite number of iterations. In a latter case, it is proved in [59] that there is a limit point amongst the datapoints computed with a function value equal to f_0 if the target value is achievable.

Though the Δ-DOGS family of schemes is (relative to other derivative-free optimization schemes) quite computationally efficient for the problem of characterizing and globally exploring (via the surrogate) a large range of nonconvex functions, it suffers from the same "curse of dimensionality" that plagues all derivative-free optimization schemes, and scales poorly with the dimension of the problem, n. The proposed hybrid algorithm mitigates this issue.

4.3 Accelerating local convergence using a derivative-based method

This section discusses the blending of the global derivative-free optimization algorithm Δ-DOGS(Z) with a local derivative-based optimization approach to accelerate the process of local refinement, and to scale better with dimension than purely derivative-free global optimization approaches. The essential idea of the new approach is two-fold: once the Δ-DOGS(Z) scheme constructs a reasonably well-sampled surrogate, the best feasible point found thus far is used to initialize a local derivative-based search. Once this derivative-based search identifies a feasible local minimum point, the value and slope of the objective function at this point are used to update the surrogate, and the derivative-free search is resumed, until a new point with an improved objective function value is found, and another derivative-free local refinement is performed, etc.

For the derivative-based component of the above-described hybrid optimization scheme, we will implement a trust region method [70] which iteratively solves the following subproblem:

$$x_k = \text{argmin } q_k(x) \quad \text{subject to} \quad x \in \Omega_k, \tag{4.4}$$

where Ω_k is a subset of B, S_k is the set of datapoints available at iteration k, z_k is the point in S_k that minimizes $f(x)$, and $q_k(x)$ is a local quadratic function constructed such that $q_k(z_k) = f(z_k)$, $\nabla q_k(z_k) = \nabla f(z_k)$, and $\nabla^2 q_k(z_k) = \nabla^2 f(z_k)$ (or, some approximation

thereof).

We now define the trust region Ω_k to be used in the derivative-based component of Algorithm 4.1. Classical trust region methods take the trust region as a sphere around z_k; this approach does not work particularly well if we want to combine the trust region method with a global optimization algorithm like Algorithm 4.1. In this paper, we thus instead define Ω_k as the Voronoi cell [78] of z_k in S_k, which is a convex, linearly-constrained region defined as follows.

Definition 11. *Around each point z_k is its constrained Voronoi cell, $V(z_k)$, consisting of all points in B that are at least as close to z_k as to any other point $x_j \in \mathbb{R}^n$. Thus,*

$$V(z_k) = \{x \in B | \, dist(x, z_k) \leq dist(x, x_j) \;\; \forall \; j \in S_k\}. \tag{4.5}$$

Taking $\Omega_k = V(z_k)$, the quadratic programming problem in (4.4) may now be rewritten as

$$x_k = \text{argmin } q_k(x) \quad \text{subject to} \quad x \in V(z_k). \tag{4.6}$$

We now present, in Algorithm 4.2, a hybrid optimization algorithm combining Algorithm 4.1 and the trust-region-based deriviative-free optimization method described above. At each iteration, either the quantization of the minimizer of the search function (4.2), or the quantization of the solution to the quadratic programming problem (4.6), is added to S_k. The first case is called a *global exploration* iteration, and the second case is called a *local refinement* iteration.

Algorithm 4.2 The new hybrid optimization algorithm to minimize $f(x)$ in the feasible domain B, leveraging a gradient-based scheme to accelerate local refinement.

0. Initialize $k = 0$, $\ell = \ell_0$, and the initial set of datapoints S_0 (confined to the grid B_ℓ), and calculate $f(x)$ for all points in S_0.

1. Denote z_k as the point in S_k which minimizes $f(x)$. Calculate $\nabla f(z_k)$, calculate or approximate $\nabla^2 f(z_k)$, generate the local quadratic function $q_k(x)$, and solve the quadratic program defined in (4.6) to obtain x_k.

2a. If (4.7) is satisfied [i.e., if $q(x_k) < \eta\,(f_0 - f(z_k)) + f(z_k)$], then determine y_k as the quantization of x_k on B_ℓ.

2b. Otherwise [i.e., if (4.7) is not satisfied], calculate or update the interpolating function $p_k(x)$ and the uncertainty function $e_k(x)$ for the points in S_k, and find the minimum of the search function (4.2), denoted $\hat{x}_k$, in B. Determine y_k as the quantization of $\hat{x}_k$ on B_ℓ.

3a. If $y_k \notin S_k$, take $S_{k+1} = S_k \cup y_k$, and calculate $f(y_k)$.

3b. Otherwise (i.e., if $y_k \in S_k$), refine the mesh, $\ell \leftarrow \ell + 1$.

4. Repeat from step 1 until convergence.

The indicator used in Algorithm 4.2 to select between global exploration and local refinement is the following:

$$q_k(x_k) < \eta\,(f_0 - f(z_k)) + f(z_k). \tag{4.7}$$

If (4.7) is satisfied, the process of local refinement at this iteration is deemed to be sufficiently promising that it might ultimately lead to a local value of $f(x) \leq f_0$, and thus a (derivative-based) local refinement step is performed; otherwise, a (derivative-free) global exploration step is performed. A parameter η with $0 < \eta \leq 1$, called the *reduction factor*, is used in this indicator function.

4.3.1 Constructing the local quadratic model

We now discuss the construction of the local quadratic function $q_k(x)$. The approach used is based on Quasi-Newton methods, which construct a locally quadratic model of the objective function

$$q_k(x) = f(z_k) + \nabla f(z_k)^T (x - z_k) + \frac{1}{2}(x - z_k)^T H_k (x - z_k), \tag{4.8}$$

with a Hessian H_k approximated based on recent gradient computations; this approach can ultimately result in an algorithm with superlinear convergence. In the present work, we use the venerable BFGS method [70] for the construction of H_k. In the implementation of the BFGS method, the matrix H_k is initialized by the identity matrix, and at each iteration that a point y_k is obtained such that $f(y_k) \leq f(z_k)$, the matrix H_k is updated as follows:

$$H_{k+1} = H_k + \begin{cases} \dfrac{\gamma_k^T d_k}{\gamma_k^T \gamma_k} - \dfrac{d_k^T H_k^T H_k d_k}{d_k^T H_k d_k} & \text{if } \gamma_k^T d_k > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{4.9a}$$

$$d_k = y_k - z_k, \quad \gamma_k = \nabla f(y_k) - \nabla f(z_k). \tag{4.9b}$$

4.4 Analysis

In this section, we analyze the convergence properties of Algorithm 4.2. Under the appropriate assumptions, we will establish two main properties:

1. If the target value is achievable, the algorithm will either (a) find a feasible point with objective function value less than or equal to f_0 in a finite number of iterations, or (b) if an infinite sequence of points is generated, there will be a limit point amongst the datapoints computed with a function value equal to f_0. This property is called *target achievability*.

2. The algorithm will converge to a KKT point for the objective function $f(x)$. This property is called *local minimum convergence*.

It is established in [59] that Algorithm 4.1 has the target achievability property; however, Algorithm 4.1 does not guarantee local minimum convergence. We will establish both properties for Algorithm 4.2, subject to the following assumptions on the objective function $f(x)$ and the interpolant $p^k(x)$:

Assumption 7. *The interpolating function $p^k(x)$, objective function $f(x)$, and $p^k(x) - f(x)$ are Lipschitz with the same Lipschitz constant $\hat{L}$ in B.*

Assumption 8. *A constant $\hat{K} > 0$ exists for which*

$$\nabla^2\{f(x) - p^k(x)\} + 2\,\hat{K}\,I > 0, \quad \forall x \in B \text{ and } k > 0, \tag{4.10}$$

$$\nabla^2\{p^k(x)\} - 2\,\hat{K}\,I < 0, \quad \forall x \in B \text{ and } k > 0, \tag{4.11}$$

$$\nabla^2\{f(x)\} - 2\,\hat{K}\,I < 0, \ \forall x \in B. \tag{4.12}$$

Moreover, the gradient of $f(x)$ is Lipschitz with constant K.

Assumption 9. *The local quadratic function $q_k(x)$ and its derivative $\nabla q_k(x)$ are Lipschitz with constant $\hat{L}$ inside B.*

4.4.1 Establishing target achievability of Algorithm 4.2

By construction, each step of Algorithm 4.2 is either a local refinement step or a global exploration step. For each mesh refinement iteration of Algorithm 4.2, there are two possible cases:

1. Condition (4.7) is satisfied, or

2. Condition (4.7) is not satisfied, but y_k [the quantizer of the minimizer of $s_k(x)$] is located in S_k.

It is noted in §5 of [59] that, if an infinite number of steps are taken, then an infinite number of mesh refinement steps are taken; there are thus either an infinite number of mesh refinement steps of the first type above, or an infinite number of mesh refinement steps of the second type above (or, both). Also, by §5 of [59], if there are an infinite number of mesh refinement iterations that are of the second type above, then Algorithm 4.2 converges to a point such that $f(x) \leq f_0$.

We will now show target achievability when there are an infinite number of mesh refinement steps that satisfy (4.7).

Theorem 5. *If there is an infinite number of iterations k in Algorithm 4.2 which are*

mesh refinement and satisfy (4.7)*, then*

$$\lim_{k\to\infty} f(z_k) \leq f_0. \tag{4.13}$$

Proof. Let us consider k an iteration of Algorithm 4.2, which is a local refinement step and and also mesh refining. Then

$$f(z_k) - q(x_k) \geq \eta(f(z_k) - f_0). \tag{4.14}$$

Since $q(z_k) = f(z_k)$, and $q(x)$ is Lipschitz with constant $\hat{L}$, then

$$f(z_k) - f_0 \leq \frac{1}{\eta}\hat{L}\|z_k - x_k\|, \tag{4.15}$$

On the other hand, step k is mesh refinement. Thus, the quantizer of x_k is in S_k. However, by construction y_k is in the Voronoi cell of z_k. Therefore, z_k is a quantizer of x_k, and

$$f(z_k) - f_0 \leq \frac{1}{\eta}\hat{L}\delta_{\ell_k},$$

where δ_{ℓ_k} is the maximum discretization error at iteration k. Since there is an infinite number iterations like k, (4.13) is shown. □

We have thus established that Algorithm 4.2 will achieve the target value. Moreover, if at one iteration we achieve the target value, then all remaining iterations are local refinement iterations. In the next section, we establish the local minimum convergence

of Algorithm 4.2.

4.4.2 Establishing local minimum convergence of Algorithm 4.2

We first make a few useful definitions.

Definition 12. *Define x_k as the solution of the quadratic programming problem* (4.6) *at iteration k. There are two possible types of binding constraints at x_k:*

a. *Constraints on the feasible domain B. These constraints are called domain-sharing active constraints.*

b. *Constraints on the Voronoi cell of z_k. These constraints are called Voronoi-sharing active constraints.*

Definition 13. *Let us consider $\mathbb{S} = \{V_0, V_1, V_2, \ldots, V_r\}$ as an affinely independent*[1] *subset of the vertices of a unit n-dimensional hypercube. Then we construct a matrix A as a matrix whose i'th column is $a_i = (V_i - V_0)/\|V_i - V_0\|$. By construction, A is nonsingular. Then, the hypercube scaling factor ρ is defined as the inverse of the minimum possible value for $\sigma_{min}(A)$ (the minimum singular value of A) over all possible subsets of $\mathbb{S}$.*

Note that, for each $z \in range(A)$ such that $\|z\| = 1$, there is a unique vector $\alpha \in R^r$, such that

$$A\alpha = z, \quad \|\alpha\| \leq \rho, \quad \sum_{i=1}^{r} |\alpha_i| = \sqrt{r}\rho \leq \sqrt{n}\rho.$$

[1] s=$\{s_0, s_1, \ldots, s_d\}$ is affinely independent if $\{s_1 - s_0, \ldots, s_d - s_0\}$ are linearly independent.

Lemma 8. *Let us consider k as an iteration of Algorithm 4.2 which is a mesh refinement; then*

1. *Domain-sharing and Voronoi-sharing constraints are orthogonal.*

2. *Consider a as the normal vector of a Voronoi-sharing active constraint; then*

$$|a^T \nabla f(x)| \leq 2\,\hat{K}\delta_{L_k}, \tag{4.16}$$

   where δ_{L_k} is the maximum discretization error at step k.

3. *Consider b as an outward-facing normal vector of a Domain-sharing active constraints; then*

$$b^T \nabla f(x) \geq -\hat{K}\,\delta_{L_k}. \tag{4.17}$$

4. *Consider c as a normal vector which is perpendicular to all active constraints at x_k; then*

$$|c^T \nabla f(x)| \leq \hat{K}\,\delta_{L_k}. \tag{4.18}$$

5. *Consider d as a unit vector which is parallel to the Domain-sharing active constraints at x_k; then*

$$|d^T \nabla f(x)| \leq (1 + \sqrt{n}\,\rho)\,\hat{K}\,\delta_{L_k}, \tag{4.19}$$

   where ρ is the scaling factor of the unit hypercube.

Proof. We first show Property 1. Let us consider H_1 as a boundary of a Voronoi-sharing active constraints, then there is a point $w_k \in S_K$, such that $\|x_k - z_k\| = \|w_k - z_k\|$. By

construction, the vector $z_k - w_k$ is orthogonal to H_1. Since step k is a mesh refinement, both z_k and w_k are quantizers of x_k. As a result, according to the construction of the Cartesian grid [59], all domain-sharing active constraints like H_2 are active at both w_k and z_k. Thus, w_k and z_k lie on the boundary of H_2, which establishes Property 1.

To show Property 2, we demonstrate (4.16) is valid, where a is the normal vector of H_1. According to the mean value theorem, there is a point ξ on the line between z_k and w_k such that

$$\frac{\nabla f(\xi)^T (w_k - z_k)}{\|w_k - z_k\|} = \frac{f(w_k) - f(z_k)}{\|w_k - z_k\|}. \tag{4.20}$$

Since z_k has the minimum objective value in S_K, then $f(w_k) \geq f(z_k)$. Thus,

$$\frac{\nabla f(\xi)^T (w_k - z_k)}{\|w_k - z_k\|} \geq 0. \tag{4.21}$$

Moreover, the function $\nabla f(x)$ is Lipschitz; thus,

$$\frac{\nabla f(z_k)^T (w_k - z_k)}{\|w_k - z_k\|} \geq -\hat{K}\|z_k - w_k\| \geq -2\hat{K}\|z_k - x_k\|. \tag{4.22}$$

On the other hand, x_k is the solution of the quadratic programming problem (4.6), which is on the constraint H. Moreover, z_k and w_k are infeasible and feasible, respectively, with respect to this constraint. Further, $(w_k - z_k)/\|w_k - z_k\|$ is normal to the boundary of H_1, and goes out of the Voronoi cell. As a result,

$$\frac{\nabla q_k(z_k)^T (w_k - z_k)}{\|w_k - z_k\|} \leq 0. \tag{4.23}$$

Since $\nabla q_k(x)$ is Lipschtiz with constant $\hat{K}$, and $\nabla f(z_k) = \nabla q_k(z_k)$, it follows that

$$\frac{\nabla f(z_k)^T (w_k - z_k)}{\|w_k - z_k\|} \leq \hat{K} \|z_k - x_k\|. \tag{4.24}$$

Since iteration k is mesh decreasing, Property 2 is established.

To show Property 3, consider b as an outward-facing normal vector of a domain-sharing active constraint H_2. Since x_k is the solution of the quadratic programming (4.6),

$$b^T \nabla q_k(x_k) \leq 0. \tag{4.25}$$

Since $\nabla q_k(z_k) = \nabla f(z_k)$ and $\nabla q_k(x)$ is Lipschitz with constant K, Property 3 is established.

To show Property 4, since x_k is the solution of the quadratic programming (4.6), then

$$c^T \nabla q_k(x_k) = 0, \tag{4.26}$$

Similarly, since $\nabla q_k(z_k) = \nabla f(x_k)$, and $\nabla q_k(x)$ is Lipschitz with constant $\hat{K}$, Property 4 is established.

Finally we consider Property 5. By construction, d can be written as

$$d = d_1 + d_2 \quad \text{where} \quad d_1 = \sum_{i=1}^{r} \alpha_i a_i, \tag{4.27}$$

where a_i are the normal vectors of the Voronoi-sharing active constraints, and d_2 is

a vector which is perpendicular to the domain-sharing active constraints at y_k. Using (4.16) and (4.18), and the triangular inequality, we have:

$$|d^T \nabla f(z_k)| \leq \delta_{L_k}[2\,\hat{K} \sum_{i=1}^{r} |\alpha_i| + \hat{K}\|d_2\|].$$

Furthermore, $\|d\| = 1$, and d_1 and d_2 are orthogonal; thus, $\|d_1\| \leq 1$, $\|d_2\| \leq 1$, and

$$|d^T \nabla f(z_k)| \leq \hat{K}\delta_{L_k}[2 \sum_{i=1}^{r} |\alpha_i| + 1].$$

On the other hand, a_i is a vector normal of a boundary of the Voronoi cell of z_k. Therefore, there is a point, $w_i \in S_K$, such that $\|y_k - z_k\| = \|y_k - w_i\|$. Moreover, since iteration k is a mesh refinement, then $\{z_k, w_1, w_2, \ldots, w_r\}$ are distinct quantizers of x_k. As a result, they are located at the vertices of a hypercube. In other words, the a_i are the vectors obtained by connecting one vertex of a uniform hypercube to the other vertices; thus, $\sum_{i=1}^{r} |\alpha_i| \leq \sqrt{n}\rho$, which establishes Property 5. □

We now prove the local minimum convergence of Algorithm 4.2.

Theorem 6. *Let us consider* $\{k_1, k_2, \ldots, \}$ *the mesh decreasing steps of Algorithm 4.2: then all the limit points of the set* $T = \{z_{k_1}, z_{k_2}, \ldots\}$ *are KKT points for the optimization problem* (4.1).

Proof. Consider z as a limit point for the set T. Then there is a subset of T like $\{z_{q_1}, z_{q_2}, \ldots\}$, such that

$$\lim_{k\to\infty} z_{q_k} = z. \tag{4.28}$$

By construction, there is an open ball around z, which does not intersect any boundary of B that does not contain z. Thus, there is a k_0 such that for $k > k_0$, and z_{q_k} could lie only on the boundaries of B that include z. Furthermore, since z_{q_k} is the quantization of x_{q_k}, $A_a(y_{q_k}) \subseteq A_a(z)$, where $A_a(x)$ is the matrix whose rows are the set of active constraints at x in B. As a result, according to Lemma (8),

$$|p^T \nabla f(z_{q_k})| \leq (1 + \sqrt{n}\,\rho)\, K\, \delta_{L_{q_k}}, \forall p \in null(A_a) \tag{4.29}$$

$$p^T \nabla f(z_{q_k}) \geq -(1 + \sqrt{n}\,\rho)\, K\, \delta_{L_{q_k}}, \forall p \in row(A_a) \tag{4.30}$$

Since $\delta_{L_{q_k}}$ converges to zero, z is a KKT point. □

4.5 Results

In this section, we compare the performance of (a) the original Algorithm 4.1, with (b) Algorithm 4.2 with steepest descent applied for local refinement, and (c) Algorithm 4.2 with the BFGS formula applied for local refinement. The test function considered is the n-dimensional Styblinski Tang function, which is a standard benchmark test for global optimization:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^4 - 16\, x_i^2 + 5\, x_i}{2} - 39.16616\, n, \tag{4.31}$$

$$\text{where} \quad L = \{x | -5 \leq x_i \leq 5\}.$$

An initial grid level of $\ell_0 = 3$ is considered, and the algorithm continues until the grid level of $\ell = 8$ is terminated. Note that the optimizations are terminated when $\mathrm{Dis}(x_k, S_K) \leq 0.005$, which leads to a comparable order of accuracy for both methods (i.e. the maximum discretization error of level $\ell = 8$ is close to 0.005). The initial datapoints in S_E^0 are constructed by $n + 1$ points as follows:

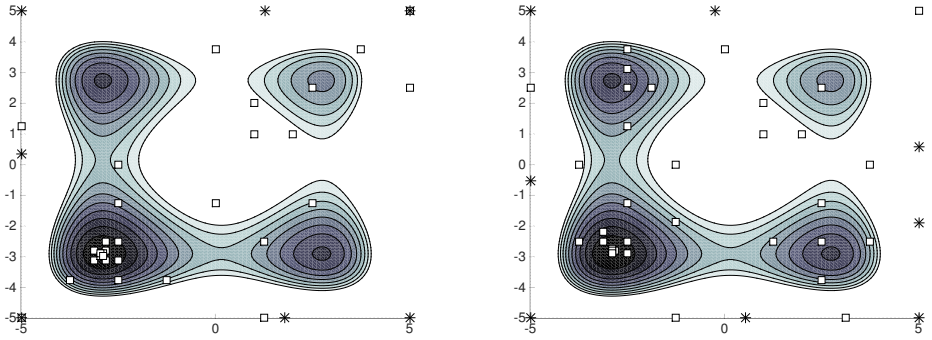
$$S_E^0 = \left\{ x_0, x_0 + \frac{b_i - a_i}{2^{\ell_0}} e_i, \forall i \in \{1, 2, \ldots, n\} \right\}. \tag{4.32}$$

For each i, e_i is the i^{th} main coordinate direction, and x_0 is an initial point on the grid of level ℓ_0. In this section, we consider two different points of x_0 for the initialization of Algorithms 4.1 and 4.2, as shown in Figs. 4.2 and 4.4.

Fig. 4.2 illustrates the position of the datapoints that are used during the optimization process for $n = 2$. With initial points $(x_0 = 0.55, x_0 + 0.2\, e_i)$ which are far from all local optima, Algorithm 4.2 focuses on global exploration; as a result, the number of function evaluations required for convergence is similar to that required by Algorithm 4.1.

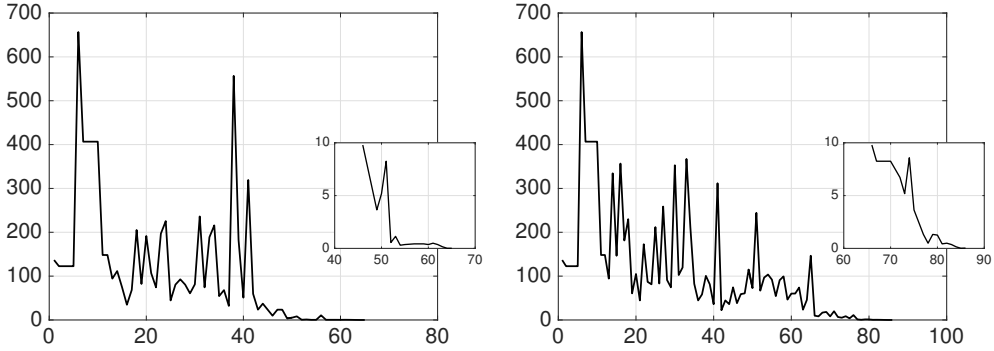
Conversely, with an initial point that lies close to a local optima, Algorithm 4.2 performs a much more efficient local refinement than Algorithm 4.1, resulting in much faster convergence. Table 4.1 reports noticeable differences that indicate significant advantages for using Algorithm 4.2 in higher dimensional problems.

As described in §3.B, the local refinement of Algorithm 4.2 can incorporate either gradient information or an approximation of the Hessian using the BFGS update



(a) Algorithm 4.2 w/ BFGS, $\eta = 0.5$. (b) Algorithm 4.1.

Figure 4.2: Exact $f_0 = f(x^*) = 0$, with $n = 2$.



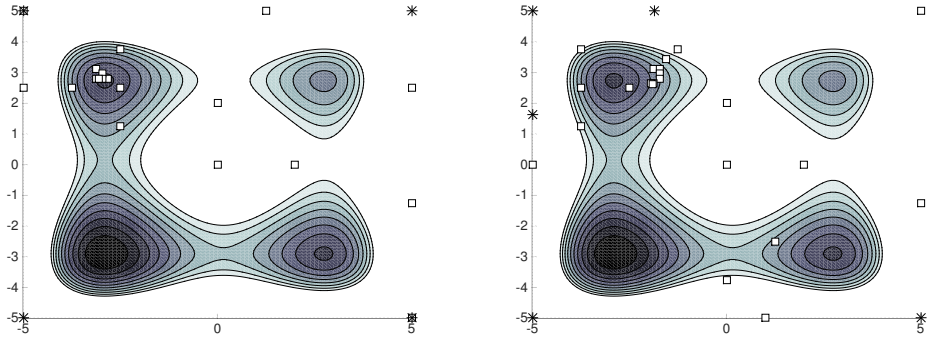
(a) Algorithm 4.2 w/ BFGS, $\eta = 0.8$. (b) Algorithm 4.1.

Figure 4.3: Exact $f_0 = f(x^*) = 0$, with $n = 4$.

Table 4.1: Algorithm 4.2 with $\eta = 0.8$, and with steepest descent and BFGS for local refinement, vs. Algorithm. 4.1. Results averaged over 5 different initial values in each case.

Average # fun. eval. / Dimension	$n = 2$	$n = 3$	$n = 4$
Algorithm 4.1 Δ-DOGS [59]	22.5	49	98.5
Algorithm 4.2 with BFGS	25	38	77.8
Algorithm 4.2 with steepest descent	27.2	61.2	59.4

formula. Table 4.1 demonstrates, as expected, that using the Hessian approximation generally has a superior convergence rate as compared with using steepest descent. Additionally, it is observed that the accuracy of the solution is significantly improved for a fixed number of function evaluations when the BFGS update formula is used. As ex-



(a) Algorithm 4.2 w/ BFGS, $\eta = 0.8$. (b) Algorithm 4.1.

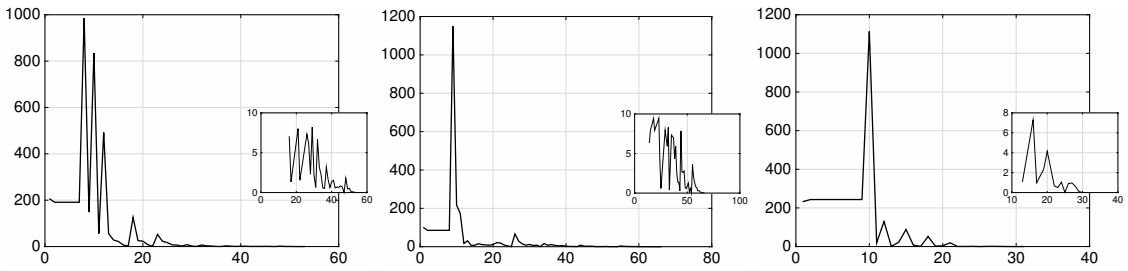
Figure 4.4: Target achievability with $f_0 = 20 > f(x^*) = 0$.

pected, in the case of Hessian approximation the grid B_{ℓ_0} is refined faster than when using gradient information only.

Note that Algorithm 4.2 with gradient descent in some situations get stuck at a local solution and performs many unnecessary function evaluations before starting to explore more globally. Due to this issue in some specific situations Algorithm 4.2 with gradient descent becomes more computationally expensive than Algorithm 4.1 and Algorithm 4.2 with BFGS.

In the case that the estimated solution, f_0, is greater than the global solution, $f(x^*)$, we see another significant advantage of Algorithm 4.2 over Algorithm 4.1. Algorithm 4.1 persists in using global search to find f_0, and stops without convergence using local refinement; thus it does not guarantee to even find a local solution when $f_0 > f(x^*)$. However, Algorithm 4.2 continues its local refinement until it converges to a KKT point. This is illustrated in Fig. 4.4.

The computation cost of Delaunay triangulation grows rapidly as the dimension n of the problem grows. By using Algorithm 4.2 with a good initial guess, the



(a) $n = 6$, $\eta = 0.8$, 80 LRS, 11 GES, 5 MRS, 7 initial points $\{x_{0,i} = 0.6, x_{0,i} + 0.1\, e_i\}$.

(b) $n = 7$, $\eta = 0.4$, 106 LRS, 11 GES, 7 MRS, 8 initial points $\{x_{0,i} = 0.13, x_{0,i} + 0.1\, e_i\}$.

(c) $n = 8$, $\eta = 0.1$, 36 LRS, 11 GES, and 8 MRS, 9 initial points $\{x_{0,i} = 0.4, x_{0,i} + 0.1\, e_i\}$.

Figure 4.5: Applications of Algorithm 4.2 with BFGS. See text.

new algorithm can converge to the global solution with a reasonable number of function evaluations even up to $n = 8$; see, e.g., Fig. 4.5. Here, we apply Algorithm 4.2 with BFGS to a variety of examples with different numbers of local refinement steps (LRS), global exploration steps (GES) and mesh refinement steps (MRS). Optimizations in these higher dimensions were simply not possible using Algorithm 4.1, due to the high computational cost of computing Delaunay triangulations in these dimensions.

Note also that η specifies a trade-off between global exploration and local refinement, and it is desired to decrease η as the dimension of the problem is increased in order to perform more local exploration rather than expensive global searches. In the case of a low-dimensional problem ($n < 6$), the performance of the algorithm is not highly sensitive to the choice of η; therefore, in most of the simulations reported here, we have taken $\eta = 0.8$. As we increased the dimensionality of the problems considered, we generally found that reducing the value of η was beneficial, in order to focus more on local refinement.

4.6 Conclusions

This paper introduces a modification to the Delaunay-based derivative-free optimization algorithm scheme Δ-DOGS(Z), as proposed in [59], accounting for gradient information in order to accelerate the local refinement iterations. The new scheme, Algorithm 4.2, has three main modifications as compared with the original Δ-DOGS(Z) algorithm:

- A criterion (4.7) for the anticipated reduction due to a local refinement step is introduced, in order to decide between taking a derivative-based local refinement step or a derivative-free global exploration step at each iteration. This criterion has an adjustable parameter η; values in the range of $\eta = 0.5$ to $\eta = 0.8$ were found to be effective.
- A new trust-region local optimization method is used, in which the trust region is characterized by the constrained Voronnoi cell of the available datapoints in the (bound) feasible domain. To guarantee local convergence of the trust-region method, all the datapoints are coordinated by a grid, with this grid being successively refined as the optimization algorithm proceeds.
- In order to accelerate the convergence of local refinement scheme and the hybrid method that uses it, Algorithm 4.2, the Hessian of objective function is approximated via the usual BFGS formula.

Proof of global convergence of the new scheme, under the appropriate assumptions, is established. Further, in the numerical experiments we have performed thus far, Algo-

rithm 4.2 is found to significantly accelerate local convergence, to handle efficiently nonconvex functions with many local minima, and to scale better with dimension than purely derivative-free global optimization approaches.

Acknowledgements

Chapter 4 contains material that has been submitted for publication and is under review. S. R. Alimo, P. Beyhaghi, T. R. Bewley, “Deluanay-based optimization algorithm via global surrogates incorporating derivative information ", submitted to *Control Decision Conference 2017*. The dissertation author was the primary researcher and author of this material.

Chapter 5

Implementation of dense lattices to accelerate Delaunay-based optimization: Δ-DOGS(Λ)

I will not say I failed 1000 times, I will say that I discovered there are 1000 ways that can cause failure. —Thomas Edison

5.1 Introduction

This chapter presents a derivative-free optimization algorithm to minimize a possibly nonconvex function with the feasible domain in a parameter space that is a simple

convex region bounded by linear inequality constraints:

$$\text{minimize } f(x) \text{ with } \; x \in L = \{x \,|\, A\, x \leq b\}, \;\; a < b. \tag{5.1}$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $A \; \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$.

Many modern optimization approaches for shape optimization of computer-aided designs converge without derivative information, and require only weak regularity conditions [5, 7, 29, 60] to solve problems in the form of (5.1). Since neither the derivative information of the objective function nor the analytical expression of $f(x)$ is usually available, derivative-free optimization is an attractive method to find an optimal solution for (5.1), and has a wide range of applicability in engineering, finance, computational chemistry, computer science, and social science. The methods are suitable for problems in which $f(x)$ is computationally/experimentally expensive to evaluate, when $f(x)$ may be locally nonsmooth, and when derivative information may be unavailable or unreliable. In derivative-free optimization, there are local and global schemes:

Local derivative-free optimization methods, like the Direct search methods [22], involve the comparison of each trial solution with the best previous solution. This class includes the most common derivative-free methods, such as General Pattern Search (GPS) [79, 80] and Mesh Adaptive Direct Search methods [24, 26, 27]. The advantage of these methods is the easy implementation, suitably for black-box optimization problems, and rigorous convergence analysis to a first-order stationary point (or to a KKT point). They do so in part by leveraging lattice theory. However, Direct search

methods can only converge to a local solution, and require many function evaluations to explore the feasible domain globally, due to the lack of an appropriate global search exploration. Most of the Direct search methods quantize the points on the Cartesian grid, but it was shown in [28] that usage of other lattices, such as different packing lattices, could enhance the convergence results.

On the other end, *global* derivative-free optimization methods seek to find a global solution of the nonconvex problem (5.1). Global optimization usually is classified into deterministic and stochastic approaches.

Stochastic approaches usually perform exhaustive random sampling in the feasible domain based on a search strategy [81]. On the other hand, Deterministic approaches can be leveraged when information exists about the objective function, making them a popular approach in simulation-based optimization. Response surface methods (RSMs) and branch and bound methods are the prime examples of deterministic approaches.

Branch and bound methods split the optimization search into a tree of subsets, and make decisions about the protectiveness of the subsets for refinement and further search by imposing some conditions on the objective function. The most popular condition for the objective function is the bound for the Lipschitz norm [82, 83]. Under appropriate conditions on the objective functions, this class of optimization methods are provably globally convergent. However, this approach overemphasizes the global search, and as a result, has issues with the speed of convergence. Moreover, in practice, implementation of such a scheme requires that a non-differential search function is minimized at each iteration, making the branch and bound methods computationally

intractable as the dimension of the problem and generated datapoints increase.

Due to such problems, RSMs are a more popular approach for shape optimization to solve (5.1). RSMs employ a surrogate model based on the available datapoints to approximate the behavior of objective functions over the feasible domain. The Surrogate Management Framework (SMF) [29] is the most well-known scheme. SMF uses a Kriging-based surrogate model for the global exploration. However, as the datapoints accumulate close to a minimizer of the problem, the behaviour of this surrogate model becomes ill-conditioned. To guarantee the convergance and increase the speed of the method, usually RSM is combined with Direct search methods.With such methods, a surrogate model is used which builds both an approximation of the function itself, $p(x)$, as well as a model of the uncertainty of this estimate, $e(x)$, over the entire feasible domain of parameter space.

Delaunay-based Derivative-free Optimization via Global Surrogates (Δ-DOGS) is a family of RSM algorithms designed for low-dimensional optimization problems, in which the objective function is both nonsmooth and expensive to evaluate [37]. The novelty of the Δ-DOGS algorithms is in modeling the surrogate function uncertainty regardless of the surrogate itself. However, the convergence to a global minimum is guaranteed if and only if the maximum circumradius of all the triangulations in the domain of solutions is bounded.

During the optimization process, Δ-DOGS models the objective function with a (computationally inexpensive) interpolating "surrogate", and represents the uncertainty of the surrogate using a synthetic piecewise-quadratic model built on the framework of

a Delaunay triangulation of the available datapoints. Unfortunately, the behavior of this artificially-generated uncertainty function is sometimes found to be somewhat irregular near the feasible domain boundaries.

This issue is addressed first in [37] by proposing *the boundary constraint projection algorithm* to guarantee the convergence and regularize the behaviour of the uncertainty function by bounding the maximum circumradius of the triangulations. Later, [59] used Δ-DOGS(Z), and showed that even by using the boundary constraint projection algorithm, the surrogate uncertainty function behaves irregularly close to the boundary of feasibility. Thus, to solve the bound simple problem, [59] focused exclusively on simple bound constraint problems, and restricted the point to lie on the Cartesian grid, which is successively refined as the algorithm converges to an optimal solution. However, this solution cannot be extended to the linear constraint problems, and the data points are restricted to only lie on the Cartesian grid points.

This chapter introduces Δ-DOGS(Λ), which generalizes Δ-DOGS(Z) to tackle linear constraint problems, similar to [37], to address more general problems. To solve such problems, we consider two search functions, a "discrete search function" (over existing datapoints) and a "continuous search function" (over the entire feasible domain), while restricting the datapoints to lie on any general grid (which coincides with the corners of the feasible domain) that is successively refined as convergence is approached. This quantization strategy effectively prevents datapoints from clustering near the boundary of the feasible domain. Moreover, this chapter considers the more difficult case with linear constraints on the feasible domain, for which the generalization of [59]

is non-trivial.

The structure of this chapter is as follows: Section 5.2 and 5.3 introduce the innovative elements that compose our quantization process, and a new approach to construct the lattice points on a linear constrained domain. Moreover, the new algorithm itself, dubbed Δ-DOGS (Λ), is described. In addition, section 5.2 analyzes the convergence properties of the new algorithm, and establishes conditions which are sufficient to guarantee its convergence to the global minimum. Section 5.4 presents the concept of acceptable quantizers, an extension of the concept of the minimum distance quantizer defined in [59] that we need for our innovative algorithm. Section 5.5 applies the new algorithm to a selection of benchmark problems in order to illustrate its behavior and performance. Some conclusions are presented in Section 5.6.

5.2 Delaunay-based optimization for linearly constraint domains

5.2.1 Initialization

In contrast to the algorithms developed in [1, 37, 58], the algorithm developed here is initialized by two different sets of points, "evaluated set" and "support set". There is no need to perform function evaluations at all of the vertices of the feasible domain L [59]; however, it is more efficient to initialize with a set of $n + 1$ points whose are interior points. These $n + 1$ points which are affinely independent are initial evaluated set.

This is useful since in many application based problems there are some prior knowledge for the position of the minimizer exist. If such a prior knowledge was not available the initial points are considered the position of the body center of the feasible domain with n perpendicular basises. On the other hand, the initial support set points are determined by the following procedure:

(A) find all of the vertices of the feasible domain L,

(B) remove all redundant constraints from the rows of $A\,x \leq b$, and

(C) project out any equality constraints implied by multiple rows of $A\,x \leq b$. In fact, we project the feasible domain onto the lower dimensional space that satisfies the equality constraints.

This procedure will be described in detail in §2 of [37]. The optimization algorithm developed in this work builds a Delaunay triangulation within the convex hull of the available points in both evaluated set and support set, and incrementally updates this Delaunay triangulation at each new datapoint (that is, at each new feasible point $x \in L$ at which $f(x)$ is computed as the iteration proceeds) [37, 84].

In the case of box constraints, [59], step (A) of the above procedure corresponds to 2^n function evaluations which are trivial to enumerate and it has been studied in [59].

In the more general case of linear constraints, (5.1), identifying the vertices of the feasible domain is slightly more involved and an algorithm is introduced in [37] to find the vertices of the domain in this case. All of these points are considered as

support points to build Deluanay triangulation over the feasible domain. We review this brute-force procedure to find all of the vertices of the feasible domain then:

1) Check the rank of all $\binom{m}{n}$ $n \times n$ linear systems that may be chosen from the $m > n$ rows of $Ax \leq b$,
2) For those linear systems in step 1 that have rank n, solve for the active set of constraints at a given point $\hat{x} \in R^n$ in parameter space, denote as $A_{a(\hat{x})}\,\hat{x} = b_{a(\hat{x})}$.
3) For each solution found in step 2, check to see if $A\hat{x} \leq b$; if this condition holds, then point $\hat{x}$ is a vertex.

5.2.2 Description of the optimization algorithm

Before presenting the algorithm some preliminary concepts we review the key elements of Delaunay-based Derivative-free Optimization via Global Surrogates, dubbed Δ-DOGS as well as the essential modification for the lattice-based version of the algorithm.

This algorithm is a global, derivative-free optimization algorithm to solve (5.1) using successive function evaluations inside a feasible domain to find the global minimum. At each iteration of the algorithm, a metric based on an interpolation of the existing function evaluations, a model of the uncertainty of this interpolation, and the target value of the optimization f_0 is used to find the best possible candidate point for the next function evaluation. In this work, the interpolating function and the uncertainty function, see Definition 11, at iteration k are denote by $p^k(x)$ and $e^k(x)$, respectively. This method can handle any well-behaved interpolation strategy which is twice differ-

entiable with bounded Hessian at all iterations. For the uncertainty function, a piecewise quadratic function is used which is nonnegative everywhere, and which goes to zero at the available datapoints.

The uncertainty function $e(x)$ has a number of properties which are established in Lemmas [2:5] of [37], as listed bellow:

a. The uncertainty function $e(x) \geq 0$ for all $x \in L$, and $e(x) = 0$ for all $x \in S$.

b. The uncertainty function $e(x)$ is continuous and Lipschitz.

c. The uncertainty function $e(x)$ is piecewise quadratic, with Hessian of $-2\,I$.

d. The uncertainty function $e(x)$ is equal to the maximum of the local uncertainty functions:

$$e(x) = \max_{1 \leq i \leq n} e_i(x). \tag{5.2}$$

Remark 11. *Consider S as a set of feasible points that includes the vertices of L as the set of unevaluated points S_U and some initial points as a set of evaluated points S_E. Consider Δ as a Delaunay triangulation [38] for S. Then, for each simplex $\Delta_i \in \Delta$, the* **local uncertainty function** *is defined as:*

$$e_i(x) = (r_i)^2 - \|x - z_i\|^2. \tag{5.3}$$

where z_i^k, and r_i^k are the circumcenter, and the circumradius of the simplex Δ_i^k. The

global uncertainty function, *defined as*

$$e(x) = e_i(x), \quad \forall x \in \Delta_i, \tag{5.4}$$

Lemma 9. *Consider $e^k(x)$ and $e^{k+1}(x)$ as the uncertainty functions defined based on the set S^k and S^{k+1} such that $S^{k+1} = S^k \cup x_k$ then $e^{k+1}(x) \leq e^k(x)$.*

Proof. Define Δ^k and Δ^{k+1} as the Delaunay triangulation of the set of points S^k, and S^{k+1}. If $x \in \Delta_r^k$ such that $\Delta_r^k \subseteq \Delta^k \cap \Delta_r^{k+1}$, then $e^{k+1}(x) = e^k(x)$ trivially. Otherwise, $x \in \Delta_j^{k+1} \notin \Delta^k$. According to the properties of the Delaunay triangulation, there is a simplex $\Delta_i^k \in \Delta^k$ which is not in Δ^{k+1} and shares n vertices like $\{V_1, \ldots, V_n\}$ with Δ_j^{k+1} and x_k is the other vertex of Δ_j^{k+1}. Define function $L(y) = e_i^k(y) - e_j^{k+1}(y)$

$$L(y) = (r_j^k)^2 - (r_i^{k+1})^2 + \|z_i^k\|^2 - \|z_j^{k+1}\|^2 - 2\Big[(y - z_j^k) - (y - z_i^{k+1})\Big]. \tag{5.5}$$

Therefore, the function $L(y)$ is linear that its minimum is located at one of its vertices. By construction $L(V_i) = 0$. Moreover, since Δ_i^k is a simplex in Δ^k which is not in Δ^{k+1}, x_k is located inside the circumesphere of Δ_i^k which leads to $e_i^k(x_k) > 0$. As a result, $L(x) \geq 0$.

Moreover, according to Lemma 4 [37], $e^k(x) \leq e_j^{k+1}(x)$ which establishes this lemma. □

Let S be a set of points in L which is partitioned into two subsets, $S = S_E \cup S_U$, each defined as follows:

- S_E: the set of evaluated points where the function values are available.
- S_U: the set of support points where the function values are not available. S_U is helpful during portioning L with triangulations.

Given the target value f_0 then the *continuous search function* at iteration k, denoted $s_c^k(x)$, is defined for all $x \in L$ such that

$$s_c^k(x) = \begin{cases} \dfrac{p^k(x) - f_0}{e^k(x)} & \text{if } p^k(x) \geq f_0, \\ p^k(x) - f_0 & \text{otherwise,} \end{cases} \tag{5.6a}$$

whereas the *discrete search function* at iteration k, denoted $s_d^k(x)$, is defined for all $x \in L$ such that

$$s_d^k(x) = \begin{cases} \dfrac{p^k(x) - f_0}{\text{Dis}\{x, S_E^k\}} & \text{if } p^k(x) \geq f_0, \\ p^k(x) - f_0 & \text{otherwise,} \end{cases} \tag{5.6b}$$

where $e^k(x)$ is the uncertainty function (see Remark 11) constructed with all the points in S^k, $p^k(x)$ is an interpolating function passing through all the points in S_E^k, and $\text{Dis}\{x, S_E^k\} = \min_{z \in S_E^k} \|x - z\|$.

Definition 14. *Let us consider x as a point in L, and S as a nonempty set of points in L, such that $z \in S$ is the closest point in S from x. The pair (x, S) is called activated if and only if $A_a(x) \subseteq A_a(z)$, where $A_a(x)$ is the set of active constraints at x.*

Remark 12. *If there are multiple points z which share the minimum distance from x in S, e.g., see Fig. 5.1 (c), then the pair (x, S) is activated if, for all such z, $A_a(x) \subseteq A_a(z)$.*

Remark 13. *If x is on the interior of L, then the pair (x, S) is activated for any nonempty set S. However, if x is on the boundary of L, the pair (x, S) may or may not activated, depending on the position of x and the points in S (see Fig. 5.1).*

5.2.3 Strawman form of algorithm

Now we have all of the tools to present the modified optimization algorithm, Δ-DOGS(Λ), to incorporate any dense lattice, and to solve problems with general linearly constrained domains. The steps of the algorithm are presented in Algorithm 5.1. Δ-DOGS(Λ) is the extended version of the Δ-DOGS(Z), optimization algorithm presented in [59], with two major improvements:

1. Instead of being limited to Cartesian grids, any dense lattices may be employed, and

2. the boundary of feasibility is generalized to linear constraint domains.

In [59], three main concepts were introduced to improve the performance of Δ-DOGS algorithm presented in [37]. In this chapter, to obtain the Algorithm 5.1 for linear constraint domain, we follow principles similar to those introduced by [59] to improve the performance of Δ-DOGS algorithm:

1. The datapoints in Algorithm 5.1 are restricted to lie on the fitted lattice that is occasionally refined as the iteration proceeds.

2. At each iteration, two different sets of points, “evaluated set” and “support set”,

Algorithm 5.1 Modified Delaunay-based optimization algorithm, Δ-DOGS(Λ), minimizing $f(x) : \mathbb{R}^n \to \mathbb{R}$ subject to $A\,x \leq b$ using f_0 as the target value.

1: Set $k = 0$ and initialize ℓ. Take the initial set of support points S_U^0 as all 2^n vertices of the feasible domain L. Choose at least $n + 1$ points on the initial grid, $n + 1$ of which are affinely independent, put them in S_E^0, and calculate $f(x)$ at each of these points.
2: Calculate (or, for $k > 0$, update) an appropriate interpolating function $p^k(x)$ through all points in S_E^k.
3: Calculate (or, for $k > 0$, update) a Delaunay triangulation Δ^k over all of the points in $S^k = S_U^k \cup S_E^k$.
4: Find x_k as the minimizer of $s_c^k(x)$ in L, and take y_k as its quantization onto the grid L_ℓ.
5: Find w_k as the minimizer of $s_d^k(x)$ in S_U^k.
6: If the pair (x_k, S^k) is not activated (see Definition 14), then: If $y_k \notin S^k$, take $S_U^{k+1} = S_U^k \cup \{y_k\}$, increment k, and repeat from 2; otherwise, take $S_U^{k+1} = S_U^k - \{y_k\}$, $S_E^{k+1} = S_E^k \cup \{y_k\}$, calculate $f(y_k)$, and increment k; if $f(y_k) > f_0$, repeat from 2, otherwise halt.
7: If $s_d^k(w_k) \leq s_d^k(x_k)$, then take $S_U^{k+1} = S_U^k - \{w_k\}$, $S_E^{k+1} = S_E^k \cup \{w_k\}$, calculate $f(w_k)$, and increment k; if $f(w_k) > f_0$, repeat from 2, otherwise halt.
8: If $y_k \notin S_E^k$, then take $S_E^{k+1} = S_E^k \cup \{y_k\}$, calculate $f(y_k)$, and increment k; if $f(y_k) > f_0$, repeat from 2, otherwise halt.
9: Increment both ℓ and k, and repeat from 2.

are considered. Function evaluations are available only for the points in the evaluated set that mostly contains internal points.

3. Two different search functions, "continuous search" function and "discrete search" function are considered at each iteration. The continuous search function is minimized over the entire feasible domain L and the discrete search function is only minimized only over the points in the support set.

Define x_k as the minimizer of the continuous search function, $s_c^k(x)$, in L and w_k as the minimizer of the discrete search function $s_d^k(x)$ in S_U^k. Furthermore, define y_k as the quantization of x_k onto the grid L_ℓ.

There are four possible cases at each iteration of Algorithm 5.1:

(1) The pair (x_k, S^k) is not activated (see Fig. 2a). This is called the *inactivated step*, then either:

   (a) y_k is a new point and simply added to S_U^k, and no function evaluation is performed, [are said to be *inactivated exploring step*] or

   (b) y_k is an already observed point in S_U^k, and $f(y_k)$ is calculated [are said to be *inactivated improving step*] and then y_k is removed form S_U^k and added to S_E^k.

   (Note that the other three steps below, in contrast, are said to be *activated*.)

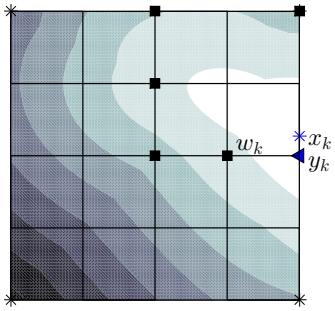
(2) The pair (x_k, S^k) is activated and $s_d^k(w_k) < s_d^k(x_k)$. This is called the *activated replacing step*: w_k is removed from S_U^k, added to S_E^k, and $f(w_k)$ calculated.

(3) The pair (x_k, S^k) is activated, $s_d^k(x_k) \leq s_d^k(w_k)$, and $y_k \notin S_E^k$. This is called the *activated improving step*: the new point y_k is added to S_E^k, and $f(y_k)$ is calculated.

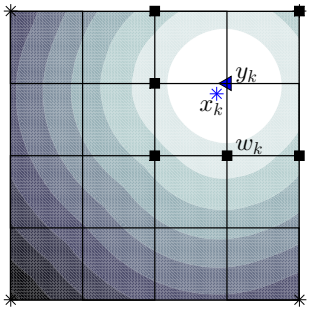
(4) The pair (x_k, S^k) is activated, $s_d^k(x_k) \leq s_d^k(w_k)$, and $y_k \in S_E^k$. This is called the *refinement step*: L_ℓ is refined, and the sets S_E^k and S_U^k are unchanged.

Note that these four possible cases are the same as steps 6:9 of the Algorithm 5.1.

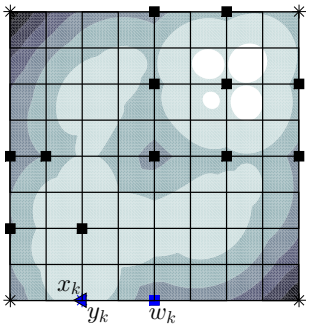
At any given iteration k of Algorithm 5.1, exactly one of the above four cases applies, and the corresponding step is taken.



(a) Inactivated step.



(b) Activated step for $x_k \in \text{interior}(L)$.



(c) Activated step for $x_k \in \partial L$, $w_k \in S_E^k, S_U^k$.

Figure 5.1: Activated and inactivated steps for different situations. Squares: S_E^k, stars: S_U^k. x_k, y_k and z_k are shown by filled square, triangle, and square respectively.

Definition 15. *Consider the reduction factor $\eta = 1$ in 4.7 and x_k as the minimizer of the $p^k(x_k)$, the surrogate model of the objective function at iteration k, then we call the iterations of Algorithm 5.1 for which $p^k(x_k) < f_0$, trust restriction iterations.*

When $p^k(x_k) << f_0$, the value of the interpolation itself may be unreliable near x_k [59]. In this scenario, it was proposed in [59] to evaluate a point x, such that $p^k(x) = f_0$ (which is, thus, closer to the existing datapoints), and evaluate the function at x instead. This approach is similar to the *trust region* approach in derivative-based optimization (see, e.g., [85]).

We accomplish this at the end of step 4 of Algorithm 5.1: if $p^k(x_k) < f_0$, a trust restriction iteration, then a point x_c is identified as the closest point in S_E^k to x_k (since the algorithm has not yet terminated, $p^k(x_c) > f_0$). Afterward a point $\hat{x}$ is found on the segment between x_c and x_k such that $p^k(\hat{x}) = f_0$. Finding $\hat{x}$ in one-dimensional space is similar to the root finding problem. For higher dimensions in [59], a false position method is proposed to find $\hat{x}$ such that $p^k(\hat{x}) = f_0$.

5.2.4 Convergence analysis

In this section, we provide the convergence proof to a global minimizer for Algorithm 5.1. However, most of the analysis will be referred from §4 in [59].

We now analyze the convergence properties of Algorithm 5.1. If the algorithm terminates after finite number of iterations k, then a point x_k is found for which the function value is less than or equal to the target value f_0; otherwise, all computed values

of the objective function are greater than the target value. In this section, we will show, in the latter case, that a limit point of the datapoints that are obtained in the evaluation set S_E includes a feasible point whose objective function is equal to the target value. Therefore, for this analysis, we will assume that Algorithm 5.1 proceeds for an infinite number of iterations.

Before analyzing the convergence of Algorithm 5.1, we first show that Algorithm 5.1 includes an infinite number of mesh refinements. To show this, a preliminary lemma is first established.

Remark 14. *If iteration k is an inactivated iteration of Algorithm 5.1; then, either* $T(x_k) = y_k \notin S^k$ *or* $T(x_k) = y_k \in S_U^k$. *It was established in Lemma 1 of [59] that by using the nearest neighbor quantizer on the Cartesian lattice in the simple bound domain that if a step is inactivated, then* $y_k \notin S^k$. *However, this lemma is not valid for the acceptable quantizer (see Definition 20) on the fitted lattice (see Definition 16) in the linearly constrained domain. This is one of the differences between the Δ-DOGS(Z) and the Algorithm 5.1.*

Since acceptable quantizer, $T(x_k)$, is not the nearest neighbor quantizer, if the pair (x_k, S^k) is *inactivated step*, then considering $y_k = T(x_k)$ either:

(a) y_k is a new point, or

(b) y_k is an already observed point in S_U^k.

We resolved resolved this problem in the way that: if (a) is true then y_k is simply added

to S_U^k, and we do not perform function evaluation; and if (b) is true, then we calculate $f(y_k)$ and add it to S_E^k.

Theorem 7. *There are an infinite number of mesh refinement iterations if Algorithm 5.1 proceeds without terminating.*

Proof. This theorem is established by contradiction. Assume that there are a finite number of mesh refinement iterations as Algorithm 5.1 proceeds, then all datapoints must lie on a grid with some level ℓ. At each iteration of Algorithm 5.1,

- if it is activated improving, then $|S_E^k|$ and $|S^k|$ are both incremented by one.
- If it is activated replacing, then $|S_E^k|$ is incremented by one and $|S^k|$ is fixed.
- If it is inactivated improving, then $|S_E^k|$ incremented by one and $|S^k|$ is fixed.
- If it is inactivated exploring, then $|S_E^k|$ is fixed and $|S_E^k|$ is incremented by one.

Therefore, at each iteration of the algorithm which is not mesh refinement, we will increment the value of $|S^k| + |S_E^k|$ by at least one. Since the number of points on the grid of level ℓ is finite, we must have only finite number of iterations which are not mesh refinements, which is in contradiction with the fact that there are infinite number of iterations for Algorithm 5.1. □

We now analyze the convergence of Algorithm 5.1. To do this, the following conditions are imposed for the objective and interpolating functions.

Assumption 10. *The interpolating functions $p^k(x)$, objective function $f(x)$, and $p^k(x) - f(x)$ are Lipschitz with the same Lipschitz constant $\hat{L}$.*

Assumption 11. *A constant $\hat{K} > 0$ exists for which*

$$\nabla^2\{f(x) - p^k(x)\} + 2\,\hat{K}\,I > 0, \quad \forall x \in L \text{ and } k > 0, \tag{5.7}$$

$$\nabla^2\{p^k(x)\} - 2\,\hat{K}\,I < 0, \quad \forall x \in L \text{ and } k > 0, \tag{5.8}$$

$$\nabla^2\{f(x)\} - 2\,\hat{K}\,I < 0, \ \forall x \in L. \tag{5.9}$$

We now establish six properties that prove convergence.

Property 6. *Consider $G(x)$ as a twice differentiable function such that $\nabla^2 G(x) - 2\,K_1 I \leq 0$, and $x^* \in L$ as a local minimizer of $G(x)$ in L. Then, for each $x \in L$ such that $A_a(x^*) \subseteq A_a(x)$, we have:*

$$G(x) - G(x^*) \leq K_1 \|x - x^*\|^2. \tag{5.10}$$

Property 7. *Consider k as an iteration of Algorithm 5.1 which is activated and a trust restriction. Then*

$$p^k(z_k) - f_0 \leq 2\,\{K + \hat{K}\}\|x_k - z_k\|^2, \tag{5.11}$$

where $K = s_c^k(x_k) > 0$.

Property 8. *Consider x^* as a global minimizer of $f(x)$ in L. Then, for each iteration of Algorithm 5.1 which is not a trust restriction, we have:*

$$\min\{\frac{s_c^k(x^*)}{\hat{K}}, \min_{z \in S_U^k}\{\frac{s_d^k(z)}{\hat{L}}\}\} \leq 2. \tag{5.12}$$

Property 9. *Consider k as a mesh refinement iteration of Algorithm* (5.1) *which is not a trust restriction. Then*

$$\min_{z \in S_E^k} f(z) - f_0 \leq \max\{3\,\hat{L}\,\delta_k, 6\,\hat{K}\,\delta_k^2\}, \tag{5.13}$$

where δ_k is the maximum discretization error of the Cartesian grid L_ℓ at this iteration.

Property 10. *If iteration k of Algorithm 5.1 is a mesh refinement and a trust restriction, then $p^k(x_k) = f_0$. Additionally, $p^k(x)$ is Lipschitz with constant $\hat{L}$; therefore,*

$$p^k(y_k) - f_0 \leq \hat{L}\|x_k - y_k\| \leq \hat{L}\delta_k. \tag{5.14}$$

Moreover, $y_k \in S_E^k$, then

$$f(y_k) - f_0 \leq \hat{L}\|x_k - y_k\| \leq \hat{L}\delta_k. \tag{5.15}$$

Remark 15. *The proof of properties 1:4 are established in Lemma 2:4 of [59].*

Lemma 10. *If Algorithm 5.1 is not terminated at any iteration, then the set $S^\infty = \lim_{k\to\infty} S^k$ has a limit point, denoted $v \in L$, such that $f(v) = f_0$.*

Proof. According to Theorem 7, there is an infinite number of mesh refinement iterations during the execution of Algorithm 5.1, denoted here $\{k_1, k_2, \dots\}$. Define v_i =

$argmin_{z \in S^{k_i}} f(z)$. Using Property 9 and Property 10, then:

$$f(v_i) - f_0 \leq \max\{3\,\hat{L}\,\delta_{k_i}, 6\,\hat{K}\,\delta_{k_i}^2\}, \tag{5.16}$$

Since Algorithm 5.1 is not terminated at any iteration, then $f(v_i) - f_0 \geq 0$. On the other hand, $\lim_{i\to\infty} \delta_{k_i} = 0$, which leads to $\lim_{i\to\infty} f(v_i) = f_0$. This proof is analogous to Theorem 2 of [59]. □

It was shown in [59, 60] that while the datapoints are restricted to the Cartesian grid, Δ-DOGS algorithms could be globally convergent under some appropriate conditions. In this section, we introduce the *fitted lattice* concept to restrict the datapoints using *acceptable quantizer* within a linear constraint domain L. In the next section, we introduce fitted lattice and acceptable quantizer to guarantee the convergence of Δ-DOGS(Λ).

5.3 Fitted lattice

In this section, we will present a new concept dubbed "fitted lattice", which is applied in Δ-DOGS(Λ) to improve the performance of the optimization algorithm Δ-DOGS(Z) developed in [59]. Δ-DOGS(Z) is based on the concept of the Cartesian grid, which is defined only for the bound constrained domain [59]. However, the Cartesian grid is not an efficient grid based on the concept of the packing density in the lattice theory, and are not the only choice for discreting parameter space. There are two key

drawbacks with Cartesian approaches for such applications. First, the discretization of space is significantly less uniform when using the Cartesian grid as opposed to the available alternatives, as measured by the packing density, the covering thickness, and the mean-squared quantization error per dimension [28, 78]. Second, configuration of the nearest-neighbor gridpoints is significantly more limited when using the Cartesian grid, as measured by the kissing number [78], which is an indicator of the degree of flexibility available when selecting from nearest-neighbor points. Therefore, we will now introduce the concept of the fitted lattice to implement other dense lattices.

Definition 16. *A sequence of set of points M_ℓ for $\ell = \{1, 2, \dots\}$ is called a fitted lattice for the feasible domain L, if and only if,*

1. *For all ℓ, M_ℓ includes all vertices of the feasible domain. Moreover, all elements of M_ℓ must be in L.*

2. *A sequence $(M_\ell)_{\ell=1}^{\infty}$ of set of points is a nested sequence, which means*

$$M_\ell \subseteq M_{\ell+1} \qquad \textit{where} \quad \forall \ell \in \{1, 2, \dots\}. \tag{5.17}$$

3. *The set $M_\infty = \lim_{\ell\to\infty} M_\ell$ should be a dense subset of L. In other words, each point in L is a limit point of M_∞.*

Remark 16. *The union of M_ℓ*

$$M_\infty = \bigcup_{\ell=1}^{\infty} M_\ell,$$

is dense in L if $\bar{M}_\infty = L$, where $\bar{M}_\infty$ is the closure of M_∞ that includes all points in M_∞ as well as all its limit points.

Remark 17. *M_ℓ icludes a finite number of points for all $\ell \in \{1, 2, \dots\}$.*

The Cartesian grids are an example of nested fitted lattices and dense series of points that hold all the above properties for a bound constrained domain.

Using the properties of fitted lattice, it is easy to show that above-mentioned properties 1:3 gets satisfied to show that Δ-DOGS is convergent to the global minimum. We now need to define a quantization process for the fitted lattice in a way to be able to satisfy other properties.

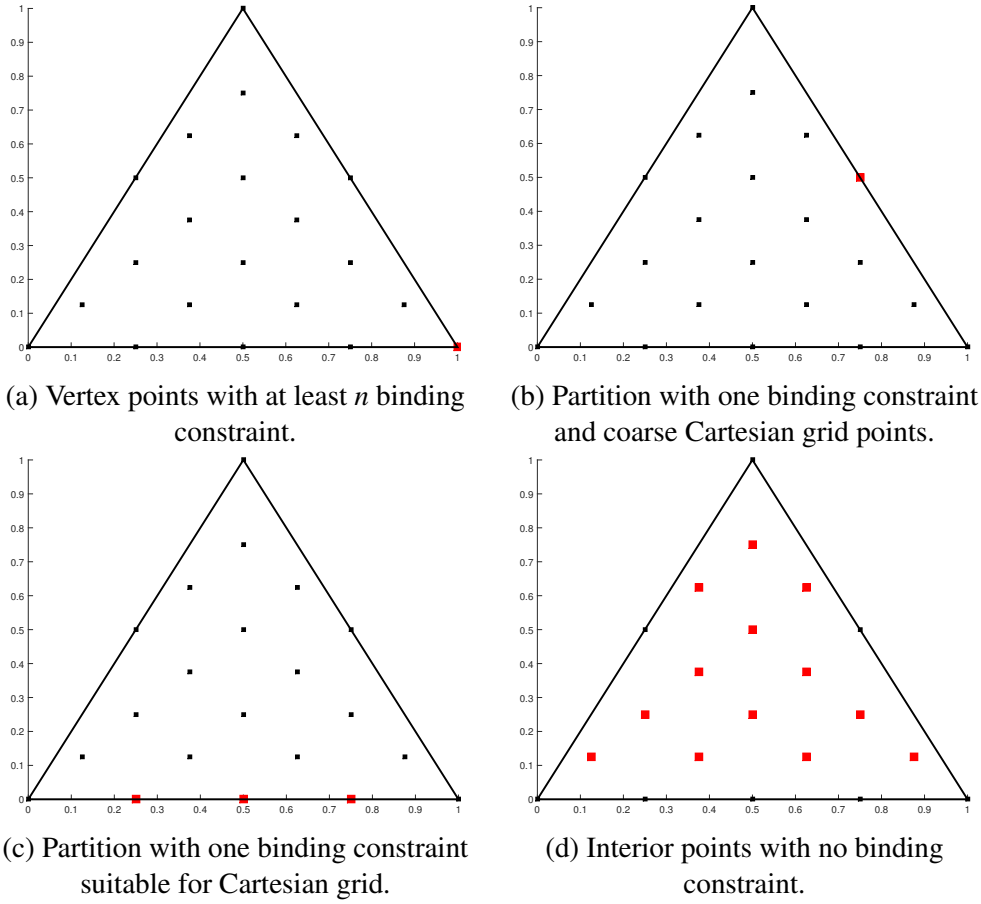
5.3.1 Constraint-based-partition

We will now develop a concept we require to construct a wide range of fitted lattices and acceptable sequences of quantizers for a linearly constraint domain *L*.

Definition 17. *Let us consider S, a set of point in L. Moreover, $P_1, P_2, \dots, P_{2^m}$ is the power set of the $\{1, 2, \dots, m\}$, where m is the number of linear constraints in L. Then the partition $S_1, S_2, \dots, S_{2^m}$ is a constraint-based partition for S with slag of $d > 0$, if $\forall i \in \{1, 2, \cdots, 2^m\}$,*

$$a_j^T x = b_j,\ \forall j \in P_i, \quad \textit{and} \quad b_j - a_j^T x \geq d,\ \forall j \notin P_i.$$

Fig. 5.2 illustrates an example of the constraint-based-partitions for a 2D prob-



(a) Vertex points with at least n binding constraint.

(b) Partition with one binding constraint and coarse Cartesian grid points.

(c) Partition with one binding constraint suitable for Cartesian grid.

(d) Interior points with no binding constraint.

Figure 5.2: Constraint partitions points (red squares) for a 2D problem with $m = 3$.

lem constrained with three linear constraints.

Another concept we need in order to define the generalize grid is the classical concept of lattice. We refer to the construction of unconstrained lattices in [28].

5.3.2 Laminated lattice

The most basic families of lattices are often referred to as root lattices due to their relation to the root systems of Lie algebra denoted $\mathbb{Z}^n$, D_n, A_n, and E_n. The basic matrix for each for these lattices are denoted by B_{Z^n}, B_{D_n}, B_{A_n}, and BE_n, respectively.

Moreover, it was observed (Fig. 5.2) that dense lattices could possibly help us increase the total number of points in each partition while the feasible constraint is defined by a set of linear constraints. However, this task is problem-dependent, and dense lattices can offer more appropriate distribution of the points on the binding constraints. See [86] for more information about dense lattices and their properties.

Any real lattice is defined simply as the *integer linear combinations* of the columns of an appropriate basis matrix B. In this work, we considered seven laminated densest lattices Λ_2 through Λ_8, are given by

We now introduce methods for quantization from an arbitrary point x in $\mathbb{R}^n$ onto its quantizaiton point, $T(x)$, on a discrete lattice, which may be defined via integer linear combination of the columns of the corresponding basis matrix B_{Λ_i}. The solution to this problem is lattice specific, and explained lattice-by-lattice in [78, 86] [1].

Remark 18. *The unconstrained quantization is defined as the minimum distance of point x from the lattice points M_ℓ at grid level ℓ.*

$$T(x) = \min_{y \in M_\ell} ||x - y|| \tag{5.18}$$

The regular quantizer of a point x with respect to the set of points in M_ℓ is calculated by rounding each element of x to the nearest point in M_ℓ.

For the unconstrained quantization process for any dense lattice see [86, 87].

[1] Note also that we neglect the problem of scaling of the lattices in this discussion, which needs to be considered.

We now develop a framework to construct a fitted lattice for the domain L based on the constraint-based partition, and a lattice with basis matrix B.

Definition 18. *The minimum distance of point x from a set of points M_ℓ, defined as*

$$Dist\{x, M_\ell\} = \min \{\|x - z_i\| \,|\, \forall z_i \in M_\ell\}. \tag{5.19}$$

The nearest neighbor quantization of a point x to M_ℓ, is $x_q = Dist\{x, M_\ell\}$. Also, the maximum quantization error (i.e., in the language of sphere packing theory, the "covering radius") of the lattice, ρ_{M_ℓ}, is defined as follows:

$$\rho_{M_\ell} = \max_{x \in M_\ell} \|x - x_q\| \tag{5.20}$$

5.3.3 Fitted lattice for linearly constrained domains

In this subsection, we will describe the construction of the fitted lattice and the quantization process for a linearly constrained domain using the properties of fitted lattice (see Definition 16) and acceptable quantizers (see Definition 21).

According to Eq. (5.1), the feasible domain L is characterized with m linear constraint functions.

We will now construct the elements of the *fitted lattice*, M_ℓ, at the grid level ℓ as a sequence of nested and dense points. Using constraint based partitioning (see Definition

17), we can decompose M_ℓ into 2^m number of subsets as follows:

$$M_\ell = \bigcup_{S \in \mathcal{P}(1:m)} M_{\ell,S} \tag{5.21}$$

where $\mathcal{P}$ the power set of {1,2,...,m} and $M_{\ell,S}$ is the fitted lattice, M_ℓ, at the grid level ℓ. By construction, each point in $M_{\ell,F}$ can be characterized as follows:

$$A_S\, x = b_S, \quad A_{\hat{S}}\, x < b_{\hat{S}},$$

where A_S include those constraints whose indices are in S, and $A_{\hat{S}}$ includes rest of the constraints. By defining U_S as the basis matrix for the null space [2] of A_S, each point x can be written as follows.

$$x = O_S + \frac{1}{N_\ell}\, U_S\, r, \quad A_{\hat{F}}[\frac{1}{N_\ell}\, U_S\, r] < b_{\hat{S}} - A_{\hat{S}}\, O_S,$$

where O_S is the projection of the origin on the space $b_S - A_S\, x = 0$, and $r \in \mathbb{R}^{ts}$, where t_S is the dimension of the null space of A_S.

Now we will construct the set $M_{\ell,S}$ such that for all $z \in \mathbb{Z}^{ts}$, then

$$x = O_S + \frac{1}{N_\ell}\, U_S\, [B_{t_S}\, z], \quad \frac{1}{N_\ell} A_{\hat{S}}\, U_S\, [B_{t_S}\, z] < b_{\hat{S}} - A_{\hat{S}}\, O_S - \epsilon_\ell, \tag{5.22}$$

where the matrix B_{t_S} is the basis matrix of t_s-dimensional space for the desired lattice

[2] U_S is a matrix whose column are an orthonormal basis for the null space of A_S.

lattice, and ε_ℓ is the pre-imposed mesh slack defined as follows.

Definition 19. *The pre-imposed mesh slack at mesh level ℓ, denoted ε_ℓ, is defined by*

$$\varepsilon_\ell = \frac{D(L)}{N_\ell}$$

where $D(L)$ is the diameter of the feasible domain L, and N_ℓ is the number of grid points in M_ℓ.

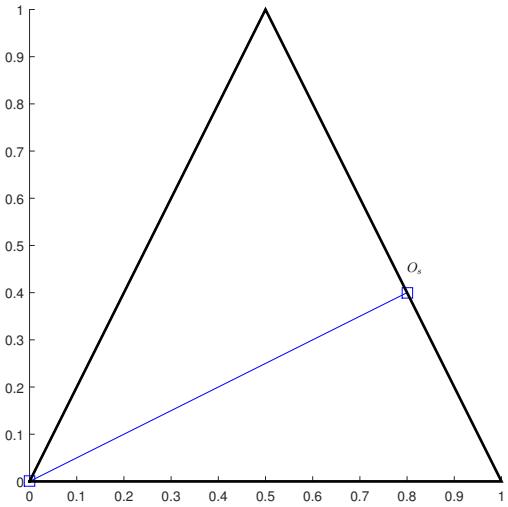
In constructing the fitted lattice on the active subspace of domain x, some points will fall too close to the vertices. These points will be removed see Fig. 5.3.

Points near the $(n-1)$-dimensional boundaries of the feasible domain are quantized to $(n-1)$-dimensional lattices defined over these boundaries; points near the $(n-2)$-dimensional "edges" of the feasible domain are quantized to $(n-2)$-dimensional lattices defined over these edges, etc. A carefully-controlled "gap" is required (and, used) between each of these families of quantization lattices in order to assure convergence.

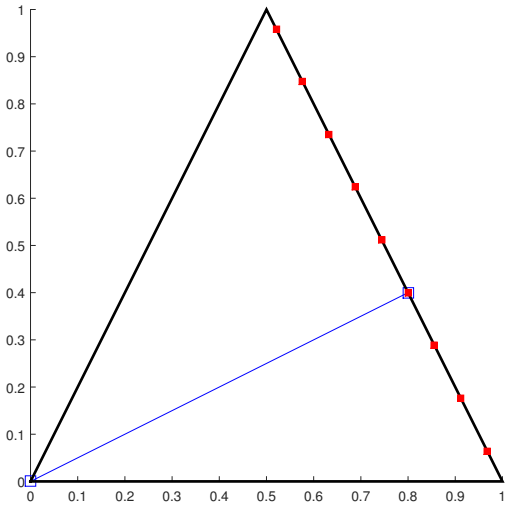
Remark 19. *It is essential to consider some special cases for $M_{\ell,S}$:*

a. *If the set $S = \emptyset$, then O_S is the origin, and U_S is equal to the Identity matrix. As a result, all unconstrained points of the mesh which are sufficiently far from the boundaries of the feasibility are in $M_{\ell,S}$.*

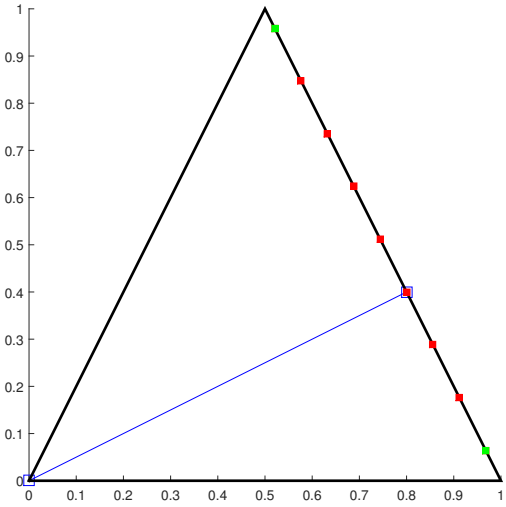
b. *If the null space of the matrix A_S is empty, then there are two possible situations: either the set $M_{\ell,S}$ is empty, or it includes exactly one of the vertices of L.*



(a) Project the origin to the active constraint as a point O_S.

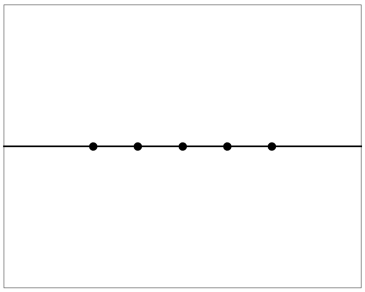


(b) At the projected domain, consider the fitted lattice.

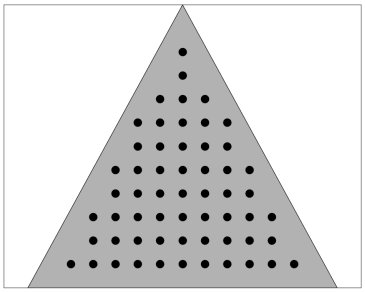


(c) Construct the fitted lattice using pre-imposed slack on a partition. The points that are located inside the gap are removed (green points).

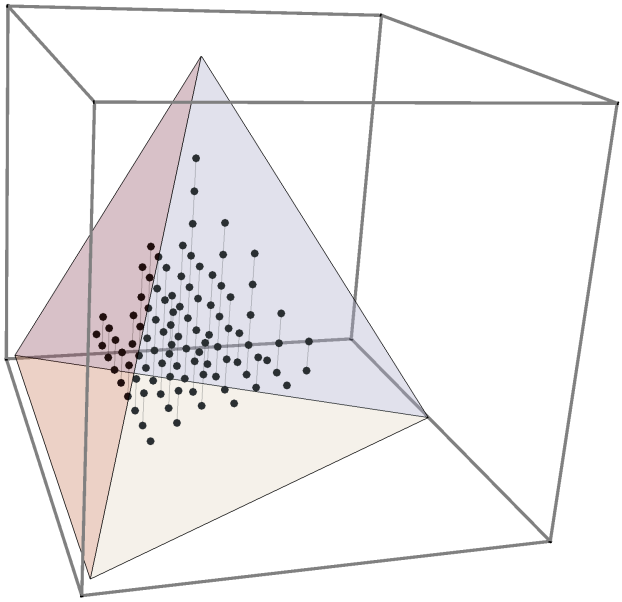
Figure 5.3: Generation of the fitted lattice on a constraint-based-partition. The green marked points get eliminated and only the red points remain in the fitted lattice.



(a) 1-dimensional case.



(b) 2-dimensional case.

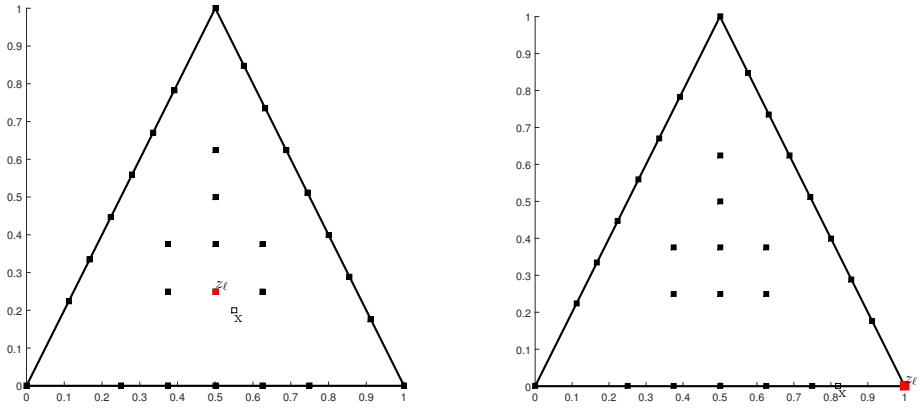


(c) 3-dimensional case.

Figure 5.4: Illustration of the interior points for a fitted lattice in 1, 2, and 3 dimensions using the Cartesian lattice as a Basis function. $M_{\ell,\emptyset} = \{O_S + \frac{1}{N} U_s B_{n-|S|}\}$,

Fig. 5.4. shows the construction of fitted lattice, $M_{\ell,S}$, on each partition. In 3D it gives a uniform pyramid, while in 2D it is illustrated by a triangulation.

We will now develop an acceptable quantizer for this fitted lattice.



(a) The quantizaiton of x on the lattice. (b) Projection into a vertex.

Figure 5.5: x and its acceptable quantizer, z_ℓ.

5.4 Acceptable quantizers

5.4.1 Acceptable quantizer

In addition to the concept of the fitted lattice, we need the concept of acceptable quantizers, which is an extension of the concept of the minimum distance quantizer defined in [59]. An acceptable quantizer can be found by Algorithm 5.2 (see section 5.4.2), and is defined as follows:

Definition 20. *A set of point $\{z_\ell\}$ is an acceptable sequence of quantizers of point x on the fitted lattice M_ℓ, if the following properties hold:*

1. *The sequence z_ℓ must be in M_ℓ.*

2. *The set of binding constraints must be preserved by the quantization process. In other words,*

$$A_a(x) \subseteq A_a(z_\ell), \tag{5.23}$$

where $A_a(.)$ is the set of active constraints of its argument.

3. *The sequence z_ℓ must converge to x.*

$$\lim_{\ell \to \infty} ||x - z_\ell|| = 0. \tag{5.24}$$

Moreover, the point z_ℓ is called the quantize of x on level ℓ.

Remark 20. *The Cartesian grid defined in [59] is a fitted lattice, and the minimum distance quantizers defined as $z_\ell = argmin_{y \in M_\ell} ||y - x||$ will construct an acceptable sequence of quantizers for the bound constraints domain.*

Remark 21. *The series of quentizers $T_i(x) = argmin_{y \in M_i} ||x - y||$ is an acceptable quantizer for the Cartesian grid. This series is typically called the nearest neighbor quantizer. This series is not acceptable for other fitted lattices, since condition 2 in Definition 20 may not be satisfied (see Fig. 5.6).*

Remark 22. *The series of function $T_i(x) : L \to M_i$, acceptable series of quantizers, have the covering radius (maximum quantization error) over the feasible domain L equal to*

$$\rho_i = \max_{x \in L} \quad ||T_i(x) - x||.$$

According to condition 1 for the series of acceptable quantizer, we have $\lim_{i \to \infty} \rho_i = 0$.

Definition 21. *The series of function $T_\ell(x) : L \to M_\ell$ are called **acceptable quantizers** for the fitted lattices M_ℓ, if they have the following properties:*

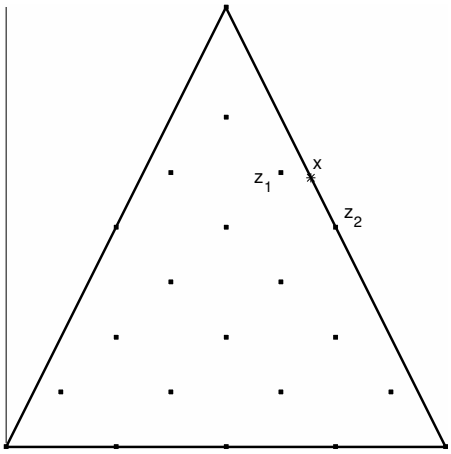


Figure 5.6: An acceptable quantizer of x needs to satisfy $\mathbf{A}_a(x) \subseteq \mathbf{A}_a(T_\ell(x))$, where $\mathbf{A}_a(z)$ is the set of active constraints at point z. In this case, z_2 is the acceptable quantizer of x even though point z_1 is closer to x.

1. *$T_\infty(x) = x$ which implies that as ℓ approaches to infinity, then $T_\ell(x)$ should uniformly converge to the point x.*

2. *If x belongs to M_ℓ, then $T_\ell(x) = x$. .*

3. *Consider $x \in L$ and $T_\ell(x) \in M_\ell$ then $\mathbf{A}_a(x) \subseteq \mathbf{A}_a(T_\ell(x))$, where $\mathbf{A}_a(z)$ is the set of active constraints at point z.*

Moreover, the covering radius (maximum quantization error) for the feasible domain L quantized onto the fitted lattice M_ℓ is defined as

$$\rho_{M_\ell} = \max_{x \in L} ||T_\ell(x) - x||,$$

Since M_ℓ is dense in L then $\lim_{\ell \to \infty} \rho_{M_\ell} = 0$.

5.4.2 Algorithm to find an acceptable quantizer

Now we want to develop the functions $T_\ell(x) : L \to M_\ell$ which is an acceptable quantizer. The primary objective for the acceptable quantizer, $T_\ell(x) : L \to M_\ell$, is that points in the interior of the fitted lattice, M_ℓ, be denser than those in regions close to the boundary of feasibility.

Remark 23. *The acceptable quantization algorithm, Algorithm 5.2, is an iterative method that is terminated either at step 2 or 4. However, it has at most* $n-1$ *number of iterations that could start from an interior point.*

Lemma 11. *The optimization problem* (5.25) *has at least one feasible solution.*

Proof. We prove this lemma by contradiction. □

Algorithm 5.2 Acceptable quantizer of $x \in \mathbb{R}^n$ on generalized mesh inside a linearly constrained domain $L = \{x | A\, x \le b\}$.

1: Find A_a as those rows of matrix A that are an active constraints at x, and b_a as the correspondent elements in the vector b.
2: If $\text{rank}(A_a) = n$, x is a vertex of L and itself is the quantizer.
3: Otherwise, set O_S as the projection of the origin on the space $A_a\, x = b_a$. Determine $U_S = \text{null}(A_a) \in \mathbb{R}^{n\times d}$, then $r = U_S^T(x - O_S)$.
4: Calculate r_q as the unconstrained quantizer of r on the laminated lattice B_ℓ^d, then set $x_q = O_S + U_S\, r_q$. If $x_q \in M_\ell$, the x_q is the quantizer.
5: Otherwise, for each constraint of L which is not active x, like $a_j^T\, x = b_j$, determine z_j as the solution of the following optimization problem:

$$\begin{aligned} z_j = \underset{y\in\mathbf{R}^n}{\text{argmin}} \quad & \|x - y\|, \\ \text{subject to} \quad & A_a\, y = b_a, \quad \text{and} \quad a_j^T\, y = b_j. \end{aligned} \tag{5.25}$$

6: Determine j that minimizes $\|z_j - x\|$, and replace x with z_j, and and repeat from 1.

5.5 Results

In this section, we will check the performance of Δ-DOGS(Λ), given Algorithm 5.1, and compare it with Δ-DOGS (as given in Algorithm 2 in [37]).

To evaluate the performance of our algorithm, the Algorithm 5.1 is applied to a representative test problem in which the objective functions $f(x)$ are (A) Styblinskia Tang function and (B) Schwefel function. To highlight the unique features of the algorithms developed, the two test optimization problems are chosen for this study, described below, have linear inequality constraints. To compare the performance of the optimization algorithms in finding a global minimum amongst several local minima, the number of function evaluations required in order to achieve a desired level of convergence is used as the evaluation criterion.

The two test problems we consider in this work, both defined such that $f(x^*) = 0$, are:

A) The solution is on the interior: a nonconvex objective function, Schwefel function, defined over an n-dimensional space, is subject to linear equality constraints such that the global minimizer is an interior point:

$$f(x) = 418.9829\,n - \sum_{i=1}^{n} x_i \sin(\sqrt{x_i}), \quad \text{where} \tag{A.1}$$

$$L = \left\{ x \in \mathbb{R}^n \,|\, 0 \le x_i \le 500\,; \quad \text{and} \quad 100\,n \le |\sum_{i=1}^{n} x_i| \right\}, \tag{A.2}$$

This problem has 4^n local minima, including the unique global minimum $x^* =$

$[420.9878, 420.9878, \ldots, 420.9878]^T$, with $f(x^*) = 0$.

B) The solution is on the boundary: A nonconvex objective function, Styblinskia Tang function, defined over an n-dimensional space, is subject to linear equality constraints such that the global minimizer is active on one of the constraints:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^4 - 16\, x_i^3 + 5\, x_i}{2} + 39.1660\, n, \quad \text{where} \tag{B.1}$$

$$L = \Big\{ x \in \mathbb{R}^n \mid -5 \leq x_i \leq 5\ ; \quad \text{and} \quad 2.0970\, n \leq |\sum_{i=1}^{n} x_i| \Big\}, \tag{B.2}$$

This problem has 2^n local minima, including the unique global minimum $x^* = [-2.907, -2.907, \cdots, -2.907]^T$, with $f(x^*) = 0$.

In this section we leverage polyharmonic spline interpolation [45, 74] as an interpolation model of the known values of the objective functions. The exact target value, i.e., $f_0 = 0$, is considered in all the problems. The effect of inaccurate target value is exclusively studied in [37, 1, 58].

We considered an initial grid level of $\ell_0 = 3$, and continued the algorithm until the grid level of $\ell = 8$ is achieved. Note that Algorithm 5.1 is terminated when $\mathrm{Dis}(x_k, S^k) \leq 0.05\, \rho_\Lambda / \rho_Z$, where ρ_Λ / ρ_Z is the maximum covering radios of dense lattices in each dimension respect to the Cartesian grid, the values are listed in Table 5.1.

$\mathrm{Dis}(x_k, S^k) \leq 0.05$, which leads to a comparable order of accuracy for both methods (i.e. the maximum quantization error of level 8 is close to 0.05).

The initial datapoints in S_E^0 are $n+1$ points as follows:

$$S_E^0 = \left\{ x^0,\ x^0 + \frac{b_i - a_i}{2^{\ell_0}}\, e^i,\quad i = 1, 2, \ldots, n \right\}. \tag{5.28}$$

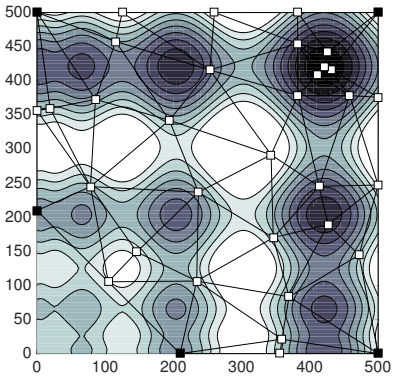
where $i = 1, \ldots, n$, e^i is one of the main coordinate directions, and x^0 is an initial point on the grid of level ℓ_0. The convergence results of Δ-DOGS(Λ) depends upon the position of the x^0; thus, two different values of x^0 are considered. For the problem (5.27) Tang test function, we take (a) $x_i^{0,a} = 0\,\forall i$, and (b) $x_i^{0,b} = -2\,\forall i$. For the problem A test function, we take (a) $x_i^{0,a} = 100\,\forall i$, and (b) $x_i^{0,b} = 400\,\forall i$.

We first considered the problem A and the problem (5.27) in $n = 2$ dimension. It is observed that, even for a bad initial point for Δ-DOGS(Λ), the convergence to the global minimum is achieved with fewer number of function evaluations as with Δ-DOGS. The position of the datapoints that are used during the optimization process are illustrated in Fig. 5.8 for the $n = 2$ dimensional implementation.

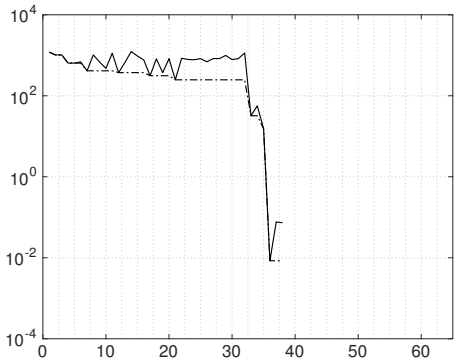
We observed that in 2D problem the original Δ-DOGS method, there were some points generated close to the boundary of the search domain (see Figure 5.8b and Figure 5.7b). In contrast, using Δ-DOGS(Λ), accumulation of the datapoints close to the boundary was not observed (see Fig. 5.8d).

Table 5.1: Summary of covering radius respect to the Cartesian lattice, $\rho_\Lambda/\rho_\mathbb{Z}$, Λ_q is the best quantizer lattice.

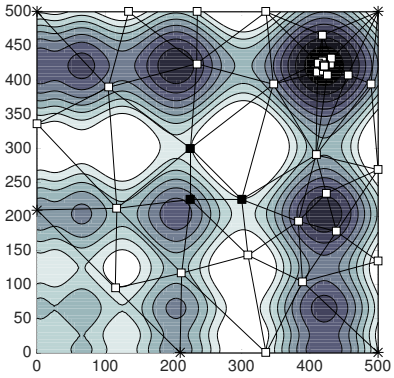
	A_2	A_3	D_4	D_5	E_6	E_7	E_8
$\rho_\Lambda/\rho_\mathbb{Z}$	1.0746	1.1225	1.1892	1.2312	1.2905	1.3459	1.4143
f_Δ	1.155	1.414	2	2.83	4.62	8	16



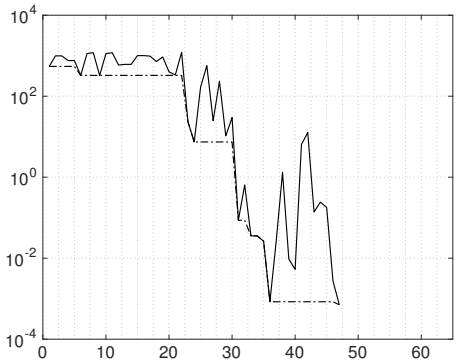
(a) Δ-DOGS



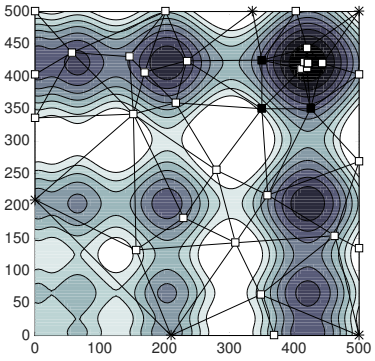
(b) All vertices,
func evals. 38



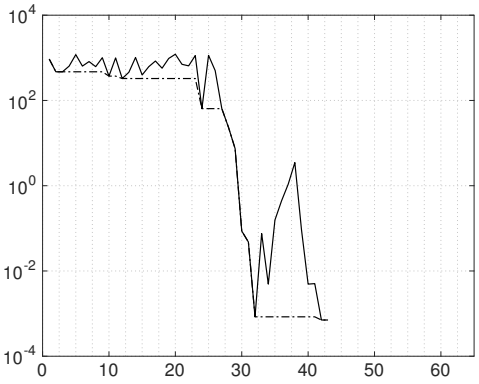
(c) Δ-DOGS(Λ)



(d) x_i=200, # func evals. 34

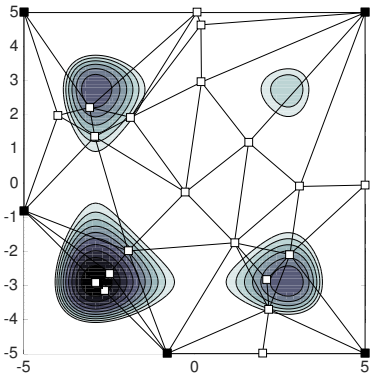


(e) Δ-DOGS(Λ)

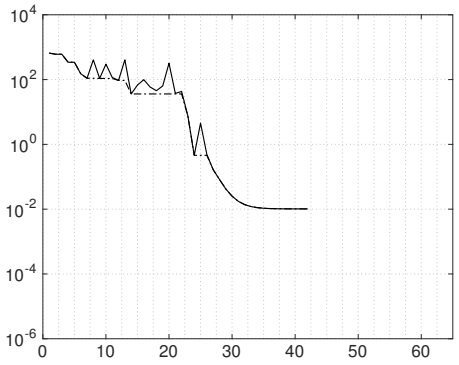


(f) x_i=350, # func evals. 43

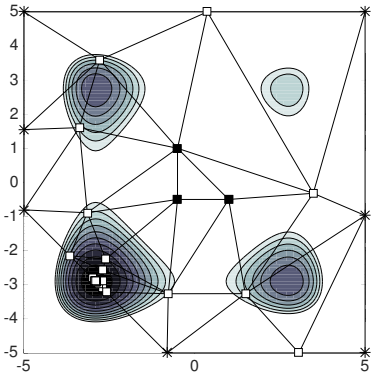
Figure 5.7: Implementation of Algorithms Δ-DOGS and Δ-DOGS(Λ) with two different initial points on problem B in 2D: (black squares) initial points, (open square) evaluated points, (stars) support points.



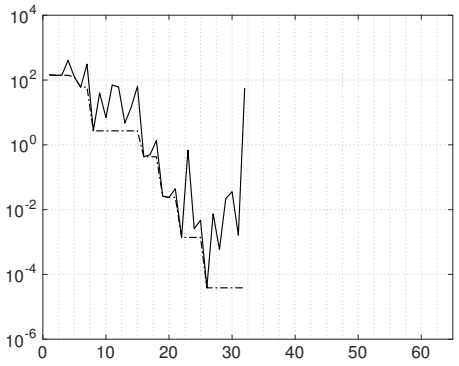
(a) Δ-DOGS



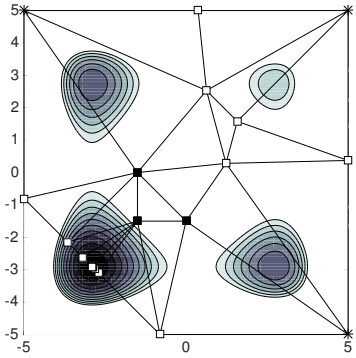
(b) All vertices,
func evals. 45



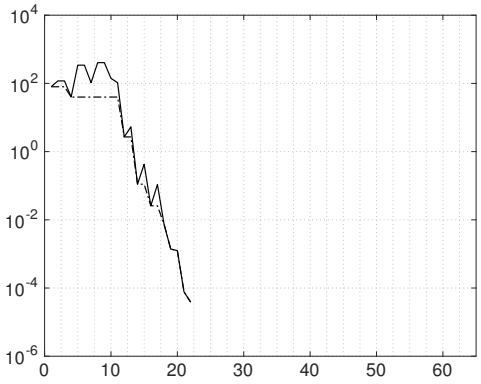
(c) Δ-DOGS(Λ)



(d) x_i=0, # func evals. 34



(e) Δ-DOGS(Λ)

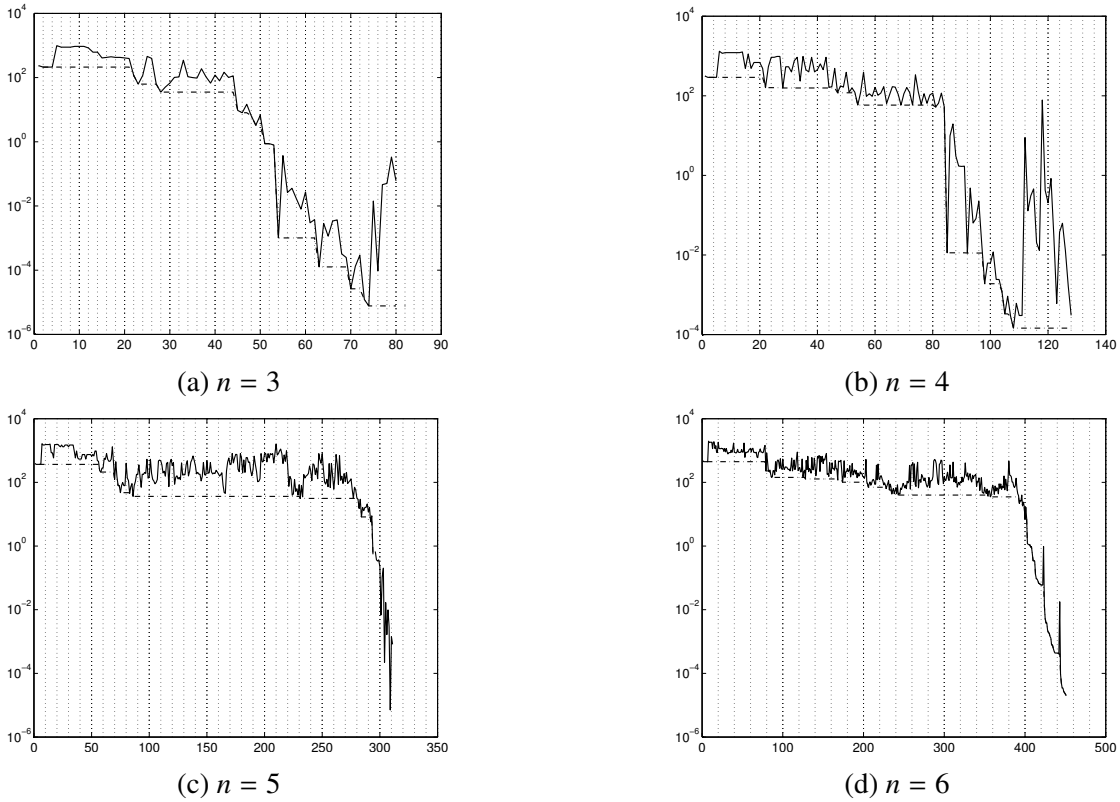


(f) x_i=-2, # func evals. 24

Figure 5.8: Implementation of Algorithms Δ-DOGS and Δ-DOGS(Λ) with two different initial points on problem (5.27) in 2D: (black squares) initial points, (open square) evaluated points, (stars) support points.

Table 5.2: Summary of the implementation of Δ-DOGS and Δ-DOGS(Λ) on problem (5.27) in $n = 2$, 3, and 4 dimensions.

Dimension	Algorithm	initial point	fn. evals	# of support points
2	Δ-DOGS	N/A	45	N/A
	Δ-DOGS(Λ)	$x_i = 0$	34	7
		$x_i = -2$	24	3
3	Δ-DOGS	N/A	88	N/A
	Δ-DOGS(Λ)	$x_i = 0$	47	18
		$x_i = -2$	113	18
4	Δ-DOGS	N/A	242	N/A
	Δ-DOGS(Λ)	$x_i = 0$	115	31
		$x_i = -2$	215	34

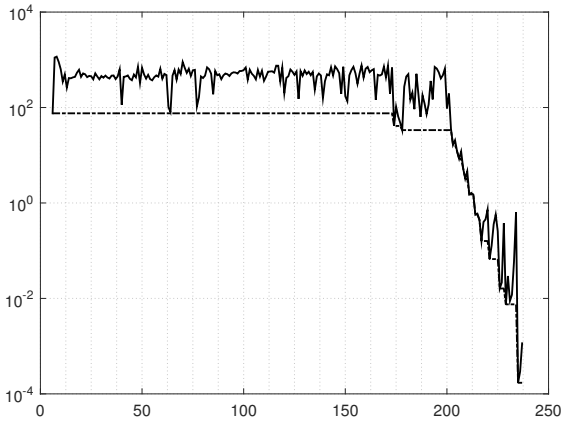


(a) $n = 3$ (b) $n = 4$

(c) $n = 5$ (d) $n = 6$

Figure 5.9: Simulation results of Algorithm 5.1 on the test problem (5.27) using the Λ basis matrix.

(a) $n = 3$



(b) $n = 4$

Figure 5.10: Simulation results of Δ-DOGS (see [1]) on the test problem (5.27).

Applying algorithms Δ-DOGS (Λ) and Δ-DOGS on the test example A for $n > 2$, it was observed that Δ-DOGS (Λ) algorithm performance was much better than that of Δ-DOGS (compare Fig. 5.9 and Fig. 5.10).

The convergence rate of Δ-DOGS was not sufficiently high; therefore, we could not go $n > 5$ for the test example. Δ-DOGS (Λ), in contrast, enabled us to reach larger "n".

It is observed that the performance of Δ-DOGS (Λ) was better than the original Δ-DOGS method introduced in [1, 37]. The main reason for this phenomena is that the introduced algorithm explores the active constraint more than the unnecessary constraints.

5.6 Conclusion

In this chapter, we have introduced Δ-DOGS(Λ), a modification of our original Delaunay-based derivative-free optimization algorithm Δ-DOGS, previously proposed in [37]. Δ-DOGS(Λ) accumulates fewer points on the boundary of feasibility, instead exploring the interior of the feasible domain more extensively. Also, we extended the algorithm Δ-DOGS(Z) developed in [59] to generalized linearly constrained problems using any general lattice.

This chapter has three main modifications as compared with the original algorithms:

- Two different sets of points are considered during the optimization process: eval-

uation points and support points. The latter set helps to regulate the triangulation developed.

- Since the uncertainty function is zero at some points that are not in the evaluation set, another metric for the search function is used at these points.
- The datapoints that are used in the Δ-DOGS(Λ) all lie on a fitted lattice that is successively refined as the iterations proceed.

As with our original algorithms, and any other derivative-free optimization algorithm, there is a curse of dimensionality, and optimization in only moderate-dimensional problems (i.e., $n \lesssim 8$) is expected to be numerically tractable. A key bottleneck of the present algorithm as the dimension of the problem is increased is the overhead associated with the enumeration of the triangulation. Another limitation of the algorithm presented in [59] is its restriction to bound-constrained domains. Note that the proposed Delaunay-based optimization algorithm, dubbed Δ-DOGS (Λ), can handle any linearly-constrained domain. Also, Δ-DOGS (Λ) uses a fitted lattice with the best option for discretization as the dimension n is increased (for further explanation, see [28]). In the next chapter, Δ-DOGS(Λ) is applied to the design of hydrofoil to maximize the efficiency of the foil.

Acknowledgements

Chapter 5 is currently being prepared for submission for publication. S. R. Alimo, P. Beyhaghi, T. R. Bewley, “Implementation of dense lattices to accelerate

Delaunay-based optimization.", in preparation to submit for *Optimization Methods and Software Journal*. The dissertation author was the primary researcher and author of this material.

Chapter 6

Delaunay-based optimization in CFD leveraging multivariate adaptive polyharmonic splines (MAPS)

I have been impressed with the urgency of doing. Knowing is not enough; we must apply. Being willing is not enough; we must do.—Leonardo da Vinci

6.1 Introduction

Factors contributing to the choice of an algorithm for optimizing a function $f(x)$ include the cost of computing $f(x)$, the local smoothness of $f(x)$, the availability of derivative information, the number of design parameters, and the shape of the feasible domain considered in parameter space. This paper considers simulation-based opti-

mization problems for which the cost of computing $f(x)$ is high, $f(x)$ may be locally nonsmooth, and derivative information may be unavailable, but the number of design parameters is relatively low (say, $n \lesssim 10$), and the feasible domain in parameter space is a simple convex region bounded by linear inequality constraints.

Over the last thirty years, as computational power has increased, derivative-free optimization algorithms have become increasingly valuable for shape optimization leveraging commercial off-the-shelf (COTS) computer-aided design (CAD) tools. Response surface methods [30] are perhaps the most computationally efficient derivative-free approaches available for shape optimization problems today, with recent applications including the design of helicopter blades and airfoils [29, 88].

Response surface methods use a computationally inexpensive model, $p(x)$, of the (computationally expensive) function of interest, $f(x)$, to approximate the trends evident in the data available at each iteration. Correlation-based interpolation models [9, 28, 36] have been widely used in such methods as surrogates to model the underlying function $f(x)$, and simultaneously to model the uncertainty associated with this surrogate. A method of this class considers the objective function as a "realization of a random process", and the parameters of the statistical model inherent to the method are tuned, using a maximum likelihood approach, to maximize the probability of the observed data at each iteration. Unlike polyharmonic spline interpolation, no metric of smoothness of the interpolant is minimized in correlation-based interpolation strategies. Thus, surrogates developed using such strategies are sometimes nonsmooth (see, e.g., the appendix of [37]), which can significantly decrease the convergence rate of the

associated optimization algorithm [67, 89, 68].

The recently-developed Delaunay-based derivative-free optimization via global surrogates (Δ-DOGS) family of methods [1, 37, 58, 59, 60, 73, 90] are response surface methods which are built upon the framework of a Delaunay triangulation of the available datapoints at each iteration. These methods are provably globally convergent under the appropriate assumptions, and are found to be remarkably computationally efficient on a range of benchmark as well as application-based problems.

As with other response surface methods, algorithms in the Δ-DOGS family iteratively minimize a search function $s(x)$ based on both an interpolation of the existing datapoints as well as a model of the uncertainty of this interpolant. Significantly, methods in the Δ-DOGS family decouple the tasks of interpolation and uncertainty modeling. A simple synthetic uncertainty model is used which is zero at each datapoint and piecewise quadratic within each simplex; this approach proves to be both effective and easy to generalize (see [1, 58]). The present paper introduces and demonstrates a new interpolation approach that is well suited for optimization algorithms in this family.

In previous implementations of optimization algorithms in the Δ-DOGS family [1, 37, 58, 59, 60, 73, 90], the polyharmonic spline (PS) interpolation approach was used. Applications included hydrofoil design optimization [2, 74]; in this particular application, though satisfactory results were ultimately achieved, a non-uniform dependence of $f(x)$ on the design parameters x over the parameter space considered was observed, as well an associated (yet, unanticipated) irregularity of the associated PS interpolants used at each iteration.

The present paper specifically addresses these shortcomings by introducing a new interpolation strategy, dubbed *multivariate adaptive polyharmonic splines (MAPS)* which, prior to performing the interpolation at each iteration, rescales the coordinate directions of the domain based on the observed variation of the available data in each direction. The hydrofoil optimization problem described in [2] represents a typical challenge problem for this effort, as its objective function $f(x)$, which characterizes the lift/drag ratio of the foil, is much more strongly dependent on some of the design parameters than others. This behavior is common in shape optimization.

The present paper specifically employs MAPS in the Delaunay-based optimization strategy developed in [73]. For comparison, Δ-DOGS with MAPS and Δ-DOGS with PS are both applied to the hydrofoil optimization problem developed in [2].

The structure of the paper is as follows: Section 6.2 briefly reviews the essential ideas of Delaunay-based optimization (Δ-DOGS) with acceleration based on Cartesian grids. Section 6.3 introduces our new interpolation strategy, MAPS, which automatically scales the parameter space prior to performing each interpolation, thereby regularizing the interpolant and, ultimately, accelerating convergence. In section 6.4, we compare the performance of the direct and the regularized approaches when searching the θ parameters of MAPS and conclude on the inadequacy of the direct approach. Section 6.5 applies the new interpolation strategy within Δ-DOGS to optimize a high-performance sailboat hydrofoil design. Some conclusions are presented in Section 6.6.

6.2 A brief review of Δ-DOGS

Algorithms in the Δ-DOGS family are already well suited for majority of low-dimensional shape optimization problems. In this paper, we consider specifically the optimization of a nonconvex objective function $f(x)$ inside a convex feasible domain bounded by linear constraints:

$$\text{minimize } f(x) \text{ with } \; x \in L = \{x \in \mathbb{R}^n | A\, x \leq b\}, \tag{6.1}$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, and the feasible domain L is assumed to be compact (closed and bounded). The compactness assumption guarantees that there is at least one solution of (6.1).

Algorithms of the Δ-DOGS family attempt to solve (6.1) using successive function evaluations at feasible points $x_k \in L$ in search of the global minimum of $f(x)$ for $x \in L$. To accomplish this efficiently, a search function $s(x)$ is minimized at each iteration; this search function is built using an interpolation of the existing datapoints, $p(x)$, a synthetic model of the uncertainty of this interpolant, $e(x)$, and a target value for the function itself, y_0.

In this work, we assume that a target value y_0 is known which is achievable, and the goal is to find a point x^* such that $f(x^*) \leq y_0$. The interpolation and the uncertainty function at each iteration k are denoted $p^k(x)$ and $e^k(x)$, respectively. The uncertainty function and the search function are defined as follows.

Definition 22. *Take S as a set of points that includes the vertices of domain L, and Δ as a Delaunay triangulation of S. The* local uncertainty function $e_i(x)$ *for each simplex $\Delta_i \in \Delta$ is defined*

$$e_i(x) = r_i^2 - \|x - Z_i\|^2, \tag{6.2}$$

where r_i and Z_i are the circumradius and circumcenter of Δ_i. The global uncertainty function $e(x)$ *is defined*

$$e(x) = e_i(x), \quad \text{for all } x \in \Delta_i. \tag{6.3}$$

The uncertainty function $e(x)$ is illustrated in Figure 6.1 in a parameter space of dimension $n = 2$. The uncertainty function $e(x)$ is characterized by the following useful properties:

1. *the uncertainty function $e(x)$ is non-negative $e(x) \geq 0$ for all points $x \in L$, and $e(x) = 0$ for all $x \in S$,*

2. *the uncertainty function $e(x)$ is continuous, Lipschitz, and piecewise quadratic.*

3. *the uncertainty function $e(x)$ is everywhere equal to the maximum of the local uncertainty functions $e_i(x)$; i.e.,*

$$e(x) = \max_{1 \leq i \leq |\Delta|} e_i(x) \quad \text{for all } x \in \Delta. \tag{6.4}$$

A number of additional useful properties of $e(x)$ are established in Lemmas [2:5] of [37].

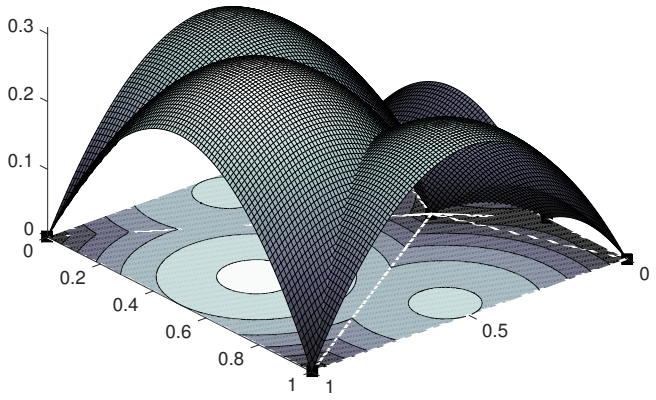


Figure 6.1: Illustration of the uncertainty function $e(x)$ in $n = 2$ dimensions.

Using the uncertainty function $e(x)$ and a suitable interpolation $p(x)$ of the available data, the search function $s(x)$ is defined as follows.

Definition 23. *Let us take S a set of datapoints that includes the vertices of the domain L, Δ as a Delaunay triangulation of S, $p(x)$ as an interpolation of the function $f(x)$ over S, and $e(x)$ as the global uncertainty function defined in* (6.3) *and built on the framework of Δ. The global search function $s(x)$ is defined as follows:*

$$s(x) = \begin{cases} \dfrac{p(x) - y_0}{e(x)}, & \text{if } p(x) \leq y_0, \\ \\ p(x) - y_0, & \text{otherwise}, \end{cases} \tag{6.5}$$

where y_0 is the target value of $f(x)$ (that is, an estimate of its lower bound).

Based on the constructions given above, the essential steps of Δ-DOGS are given in Algorithm 6.1. Figure 6.2 illustrates one iteration of Δ-DOGS on an representative

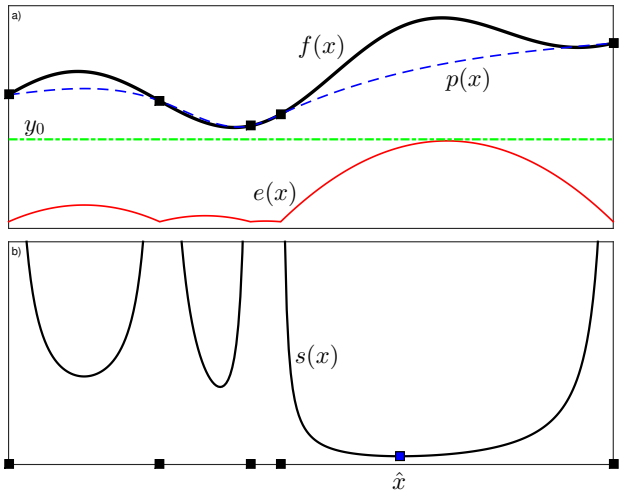


Figure 6.2: Elements of Δ-DOGS: *a*) truth function $f(x)$, interpolating surrogate function $p(x)$, model of the uncertainty $e(x)$, datapoints (black dots). *b*) search function $s(x)$, minimizer $\hat{x}$ of $s(x)$ (blue dots).

problem.

One of the challenges of the basic Δ-DOGS algorithm [37] is its overexploration of the boundaries of feasibility. This issue may be addressed by using a Cartesian grid [59] or, more generally, a dense lattice [73] to help coordinate the search, and successively refining this grid or lattice as convergence is approached. These coordination steps are useful for minimizing the accumulation of function evaluations along the boundary of the feasible domain, though they must be implemented with care. The particular variant of the Δ-DOGS optimization algorithm that is used in the present work is that described in [73], which modifies the basic Δ-DOGS algorithm to coordinate the search with lattices over L and ∂L that are successively refined as convergence is approached. The technical details of the modifications necessary to implement this idea correctly are somewhat involved, and discussed in detail in [59, 73], together with for-

Algorithm 6.1 The essential steps of Δ-DOGS.

1: Set $k = 0$. Take the set of initialization points S_0 as all M of the vertices of the feasible domain L.
2: Calculate (or, for $k > 0$, update) an appropriate interpolating function $p_k(x)$ through all points in S_k.
3: Calculate (or, for $k > 0$, update) a Delaunay triangulation Δ^k over all of the points in S_k.
4: Find x_k as a global minimizer of $s_k(x)$ in L to obtain x_k.

$$\hat{x}_k = \text{argmin}_x \; s_k(x) \quad \text{subject to } x \in L.$$

5: Calculate $f(x)$ at $\hat{x}_k$, and take $S^{k+1} = S^k \cup x_k$. Repeat from 2 until convergence.

mal proofs of convergence and illustration on model problems.

One of the parameters that is essential for accelerating the convergence of the Δ-DOGS algorithm is the estimate of the lower bound of the objective function, y_0, over the feasible domain L. It is shown in [37] that, if $y_0 \leq f(x^*)$, convergence to the global minimum is guaranteed for any twice differentiable function $f(x)$; however, values of y_0 for which $y_0 \ll f(x^*)$ tend to reduce the convergence rate. If $y_0 > f(x^*)$, the algorithm will stop at some feasible point $\tilde{x} \in L$ such that $f(\tilde{x}) \leq y_0$; in this case, convergence to the global minimum is not guaranteed.

Another important factor affecting the performance of Δ-DOGS algorithms is the choice of the interpolation strategy used. As mentioned previously, algorithms in the Δ-DOGS family can leverage any well-behaved interpolation strategy. In the previous implementations of Δ-DOGS, polyharmonic splines (PS) interpolation has been used. The ultimate performance of Δ-DOGS algorithms depends strongly on the smoothness of the interpolations used. In some situations (specifically, when the variation of the function $f(x)$ with respect to the parameters is nonuniform, with much stronger varia-

tion in some coordinate directions than others), it is has been found that PS interpolants are not sufficiently smooth. In the section that follows, we thus introduce a new interpolation method that rescales the parameter domain appropriately while performing the interpolation, thus developing a significantly smoother interpolant (and thereby, ultimately, accelerating convergence of the associated optimization algorithm).

6.3 A new polyharmonic spline interpolation algorithm for Δ-DOGS

As discussed above, the choice of the strategy to be used to construct the interpolant $p(x)$ is subtle, and strongly affects the rate of convergence of the associated optimization algorithm. Polyharmonic splines (PS) interpolation is, in general, one of the most popular interpolation strategies available today, as its formulation specifically minimizes a metric measuring the curvature of the resulting interpolant. Numerical experiments in [37] showed that PS interpolation is fairly well behaved, as compared with Kriging-based approaches, even when the available datapoints are clustered in various distinct regions of parameter space. Note also that PS interpolation has also been used in various response surface methods developed by other groups, including [68, 89].

Appropriate rescaling of parameter space is a valuable step in numerical optimization, and various recent papers have attempted to address the rescaling issue in the optimization setting. In the numerical optimization literature, there are two main ap-

proaches taken for the automatic rescaling of parameter space[1]. The first approach is to use a correlation-based model to develop the interpolant, as discussed in the third paragraph of the introduction; such approaches naturally solve for correlation length scales during the computation of the interpolant via a maximum likelihood formulation, but do not guarantee smoothness of the resulting interpolant. The second approach uses a statistical method to identify the variation of the function with respect to each parameter in the available dataset, and uses this statistical information to rescale the parameters prior to performing the interpolation at each iteration. This approach works well if the data that is used for this sensitivity analysis is well-distributed over the feasible domain; however, during the optimization processes, the dataset is expected to become clustered in some regions of the feasible domain while remaining sparse in others, which renders this rescaling approach unreliable.

In the following two sections, we first review PS, then develop and analyze our new interpolation strategy, dubbed MAPS, which includes, during the interpolation process, an automatic rescaling of the parameters based on the available data.

6.3.1 Polyharmonic spline (PS)

Polyharmonic spline interpolation is a widely-used strategy to interpolate scattered data in multiple dimensions [45]. An interpolant $p(x)$ is defined as a smooth function, which is typically inexpensive to compute, which models a "truth" function

[1]The two approaches described here may be considered specifically for the problem of dimension reduction; that is, for the exploration of $f(x)$ over a reduced number of parameters during certain steps of the optimization algorithm. This possibility will be explored in the present setting in a future paper.

$f(x)$, which might be expensive to compute, such that

$$p(x_i) = f(x_i) \quad \text{for} \quad i = 1, \ldots, N. \tag{6.6}$$

To maximize the smoothness of $p(x)$, it is suggested in [45] that the following term should be minimized

$$\int_L \|\nabla^m p(x)\|_2^2 \, dx, \tag{6.7}$$

subject to $p(x_i) = f(x_i)$, $\forall i = 1, \ldots, N$, where m is an integer such that $m \leq N$. Under these conditions, the minimizer of (6.7) gives a polyharmonic spline interpolant [45]. By choosing $m = 2$ in (6.7), the resulting interpolant will be contained in the Beppo-Livi space of distributions on $\mathbb{R}^n$ with square integrable second derivatives [91]; this choice is termed natural polyharmonic spline interpolation, and is by far the most common choice in this class for simplicity we call it polyharmonic spline (PS) interpolation.

PS may be defined as a combination of a weighted sum of a set of radial basis functions $\varphi(r)$ built around the location of each evaluation point, $\{x_i\}_{i=1}^N$, and a linear function of x:

$$p(x) = \sum_{i=1}^{N} w_i \, \varphi(\|x - x_i\|) + v^T \begin{bmatrix} 1 \\ x \end{bmatrix}. \tag{6.8}$$

The coefficients w_i and v_i are real numbers in which v_i represent the coefficients of a linear polynomial.

The following orthogonality conditions are applied:

$$\sum_{i=1}^{N} w_i = 0, \qquad \sum_{i=1}^{N} w_i \, x_{i\ell} = 0 \quad \forall\, \ell = 1, 2, \ldots, n; \tag{6.9}$$

this imposes $n + 1$ constraints, coupled with the interpolation condition (6.6), which imposes N constraints, gives the linear system (6.10) below, from which one can solve[2] for the parameters w and v in the PS interpolation formula (6.8):

$$\begin{bmatrix} F & V^T \\ V & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} f(x_i) \\ 0 \end{bmatrix} \tag{6.10a}$$

where

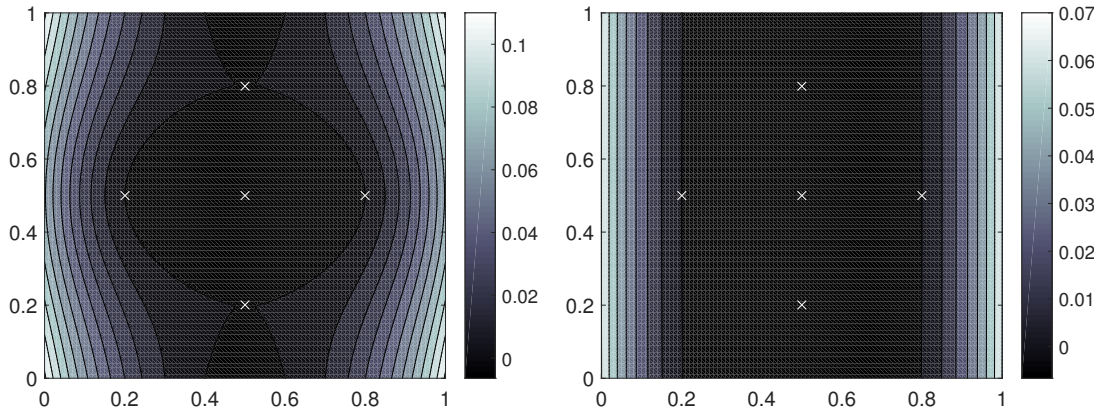
$$F_{i,j} = \varphi(\|x_i - x_j\|), \quad i, j = 1, \ldots, N, \tag{6.10b}$$

$$\text{and} \quad \varphi(r) = r^3, \quad r = \|x - x_i\|, \quad V = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_N \end{bmatrix}. \tag{6.10c}$$

The solution of the above linear system is unique [45]. Thus, by solving the linear system (6.10), the coefficients w and v are found, and $p(x)$ is determined.

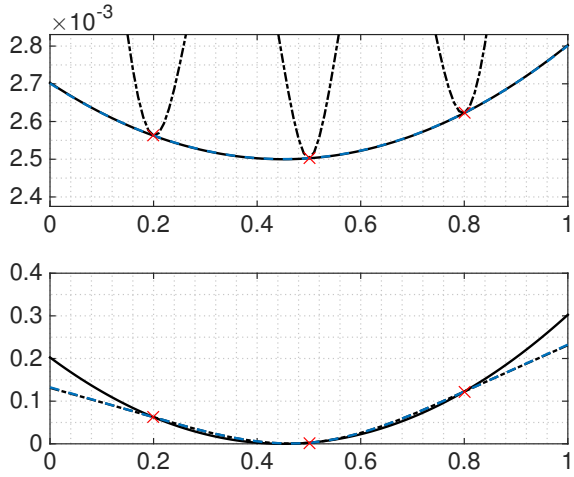
The polyharmonic spline interpolation formula in (6.8) has various shortcomings, the most significant of which is the ill conditioning of the linear system (6.10) that is solved to fit the polyharmonic spline to the datapoints. This stiffness is a direct

[2]The reader is encouraged to read about the details associated with efficiently finding these weights as described in [45].



(a) Error contour for the PS (6.8).

(b) Error contour for the MAPS (6.11).



(c) Blue dashed: MAPS interpolant. Black dashed: PS interpolant. Black solid: truth function $f(x)$. Red cross: available datapoints. Top: $f(x_1, 0.45)$ v.s. x_1. Bottom: $f(0.45, x_2)$ v.s. x_2.

Figure 6.3: PS and MAPS interpolants with respect to the truth function $f(x) = 0.01\,(x_1 - 0.45)^2 + (x_2 - 0.45)^2$ (solid). See text.

result of the function $f(x)$ having a nonuniform variation in the design parameters. The result of this stiffness is spurious oscillations of the resulting interpolant in coordinate directions with less pronounced variation of $f(x)$. Figure 6.3 illustrates this behaviour for a representative problem, and compares with the outcome of the new interpolation

strategy proposed below. Note that $f(x)$ is much more sensitive to x_2 than it is to x_1 (see variations in both directions). Using PS (6.3a) creates deviations in the x_1-direction, absent with MAPS (6.3b). Deviations reduce the convergence rate of optimization.

6.3.2 Multivariate adaptive polyharmonic splines (MAPS)

Consider now an interpolant of the form:

$$p_s(x) = \sum_{i=1}^{N} w_i \, \varphi(a.(x - x_i)) + v^T \begin{bmatrix} 1 \\ x \end{bmatrix}, \tag{6.11}$$

$$\text{where} \quad \varphi(a.r) := (a.r)^3 \quad \text{and} \quad a.r := \sum_{\ell=1}^{n} a_\ell \, r_\ell,$$

where the a_ℓ are scaling parameters, and inherent functions of w and v. The following condition is imposed:

$$\sum_{\ell=1}^{n} a_\ell^2 = n. \tag{6.12}$$

Unfortunately, the quadratic constraint (6.12) is more challenging than a linear constraint from the perspective of optimizing the weights. Thus, (6.12) is restated as a linear constraint using the change of variables $\theta_\ell = a_\ell^2$:

$$\sum_{\ell=1}^{n} \theta_\ell = n, \qquad \text{where} \qquad \theta_\ell \geq 0. \tag{6.13}$$

The formulation developed below thus works with θ instead of a. The problem of computing a MAPS interpolant thus reduces to the problem of solving for the variables

$$X = (w_1 \ldots, w_N, v_1, \ldots, v_{n+1}, \theta_1, \ldots, \theta_n)^T.$$

In contrast with the PS interpolation formula (6.8), which has $N + n + 1$ unknowns, the MAPS interpolation formula (6.11) has $N + 2n + 1$ unknowns. With the additional n degrees of freedom (i.e., the scaling parameters θ_ℓ) in MAPS, a wider range of parameters is used in order to obtain a smoother interpolant.

To improve the convergence of Δ-DOGS, we desire to use smooth interpolants at each iteration. We achieve this in the present context by performing minimum Frobenius norm (MFN) interpolations. In an MFN formulation, the Hessian of the interpolant, $\nabla^2 p_s(x)$, is minimized by minimizing the L_2-norm of the vector w [67]. The scaling θ could thus, in theory, be optimally tuned by solving

$$\begin{aligned}
\min_{w,v,\theta} \quad & \sum_{i=1}^{N} w_i^2, \\
\text{subject to} \quad & \sum_{\ell=1}^{n} \theta_\ell = n, \quad \theta_\ell \geq 0, \quad \ell = 1, 2, \ldots, n, \\
& \sum_{i=1}^{N} w_i = 0, \quad \sum_{i=1}^{N} w_i \, x_{i\ell} = 0, \quad \ell = 1, \, 2, \, \ldots, \, n, \\
& p_s(x_i) = f(x_i), \quad i = 1, 2, \ldots, N.
\end{aligned} \tag{6.14}$$

The above optimization problem is nonconvex, however, as the last constraint above is a nonlinear function of θ. For a fixed value of θ, this constraint is satisfied by solving a

linear system, and the resulting objective function may be rewritten as:

$$Q(\theta) := b^T A(\theta)^{-T} L A(\theta)^{-1} b = \sum_{i=1}^{N} w_i^2, \tag{6.15a}$$

where

$$L = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}, \quad A(\theta) = \begin{bmatrix} F(\theta) & V^T \\ V & 0 \end{bmatrix}, \qquad b = \begin{bmatrix} f(x_i) \\ 0 \end{bmatrix}, \tag{6.15b}$$

noting that

$$F(\theta)_{i,j} = \varphi\Big(\|(x_i - x_j)\|_\theta\Big), \quad i, j = 1, \ldots, N, \tag{6.15c}$$

$$\text{where} \quad \varphi(r) := r^3 \quad \text{and} \quad \|r\|_\theta := \Big(\sum_{\ell=1}^{n} \theta_\ell \, r_\ell^2\Big)^{1/2}, \quad V = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_N \end{bmatrix}. \tag{6.15d}$$

The optimum scaling parameter θ^* can thus be found using an appropriate sequential quadratic programming (SQP) solver (see, e.g., [92]), with the Hessian approximated using BFGS:

$$\theta^* = \min_{\theta \in \mathbb{R}^n} \quad Q_j(\theta) \qquad \text{subject to} \quad 1^T \theta = n, \quad \theta \geq 0. \tag{6.16}$$

The main challenge in minimizing $Q(\theta)$ is the singularity of $A(\theta)$ as one of the elements of the vector θ approaches zero. This issue is illustrated for a simple quadratic problem in Figure 6.4 (see section 6.4), which illustrates that there is a significant deviation in

the value of $Q(\theta)$ as determined via SQP applied to (6.15), when compared with the optimum value of $Q(\theta)$, as one of the scaling parameters $\theta_\ell \to 0$. It is seen that the SQP approach tends to get caught in a local minimum for small θ_ℓ. This is a common situation, especially when the actual variation of the objective function with respect to some of the parameters is small. To address this issue, a series of new problems $Q_{\lambda_j}(\theta)$ is defined and iteratively minimized as the relaxation parameter λ_j is gradually decreased towards zero:

$$Q_\lambda(\theta) = b^T \left(A_\lambda(\theta)\right)^{-T} L \left(A_\lambda(\theta)\right)^{-1} b, \quad \text{where} \quad A_\lambda(\theta) = A(\theta) + \lambda L. \tag{6.17}$$

Note that, as $\lambda \to 0$, (6.15) is recovered from (6.17). To minimize $Q_\lambda(\theta)$ efficiently, for any given λ, we need the following derivative information:

$$\frac{d\, Q_\lambda(\theta)}{d\, \theta_\ell} = 2\, B_\ell^T(\theta)\, L\, W(\theta), \tag{6.18a}$$

$$A_\lambda(\theta)\, B_\ell(\theta) = -\frac{\partial\, A_\lambda(\theta)}{\partial\, \theta_\ell}\, W(\theta), \tag{6.18b}$$

$$A_\lambda(\theta)\, W(\theta) = b \quad \text{where} \quad W(\theta) = \begin{bmatrix} w \\ \\ v \end{bmatrix}, \tag{6.18c}$$

where $A_\lambda(\theta)$ is defined in (6.17) and L, b, $A(\theta)$ are defined in (6.15b). Having this information on the derivative of $Q_\lambda(\theta)$, we may use the BFGS method to minimize it. It is known (see, e.g., [93, 94]) that such a procedure will converge to a local minimum of (6.14). As codified in Algorithm 6.2, by performing a set of relaxations towards the

Algorithm 6.2 Scheme to find the scaling parameters of MAPS (6.11)

1: Set $j = 0$. Take the set of initialization points S_0 and consider λ_0, $\theta_\ell = 1$ for all $\ell = 1, 2, \ldots, n$.
2: For $j > 0$, initialize the following system by $\theta_0 \leftarrow \theta_{\lambda_{j-1}}$ with fixed λ_j
3: Find θ_{λ_j} by solving the quadratic programming

$$\theta_{\lambda_j} = \min_{\theta \in \mathbb{R}^n} \quad Q_{\lambda_j}(\theta)$$
$$\text{subject to} \quad 1^T \theta = n, \quad 0 \leq \theta \leq n, \theta_{\lambda_j} = \text{argmin}_\theta \, [\frac{\partial W^T}{\partial \theta} (L W)]^T d_\theta$$
$$\text{subject to} \quad 1^T d_\theta = 0, \quad -n \leq d_\theta \leq n,$$

4: Update $\lambda_{j+1} \leftarrow \lambda_j/2$, increment j by one, and repeat from step 2 until (6.19) is satisfied.

solution[3], for successively smaller values of λ, it is found that reliable convergence to the desired (global) minimum of (6.14) is obtained; this optimized value of θ may then be used in the MAPS interpolation strategy (6.11).

Step 3 of Algorithm 6.2 can be solved using, e.g., the SNOPT optimization package [92], given the function of interest, $Q_{\lambda_j}(\theta)$, as in (6.17), and the gradient information, $d\, Q_{\lambda_j}(\theta)/d\, \theta_\ell$, as in (6.18), at each iteration k. The initial value of the relaxation parameter in Algorithm 6.2, λ_0, must be sufficiently large to give a well-conditioned linear system. To determine an appropriate stopping condition in this relaxation, define first the interpolation error δy_i over all points x_i for a given λ_j:

$$\delta y_i = f(x_i) - p_s(x_i; \theta_{\lambda_j}).$$

In addition, define the distance between the value of the objective function at x_i and the

[3]Note that the initial point for minimizing $Q_{\lambda_{j+1}}(\theta)$ is the minimizer of $Q_{\lambda_j}(\theta)$ (see step 2 of Algorithm 6.2).

target value y_0 as

$$\Delta y_i = f(x_i) - y_0.$$

An effective stopping criteria is reached when the error of interpolation, δy_i, reduces to, say, less than 10% of Δy_i at each datapoint x_i; that is,

$$\frac{\delta y_i}{\Delta y_i} < 0.1 \qquad \text{for all} \qquad i = 1, 2, \ldots, N. \tag{6.19}$$

After a finite number of refinements j of Algorithm 6.2, the stopping criterion (6.19) will be satisfied. An important observation is that, for the initial iterations of Δ-DOGS, since Δy_i is large, (6.19) is satisfied even for relatively large values of λ_j. As Δ-DOGS proceeds, Δy_i becomes smaller, which necessitates an increased number of refinements of λ_j to make δy_i sufficiently small to satisfy (6.19) at all points x_i, thus driving the MAPS surrogate towards a true interpolant as convergence is approached.

6.4 Comparison of direct & regularized approaches to find the θ parameters of MAPS

We now provide a brief discussion about the inadequacy of the direct approach to find θ^* when one of the scaling parameters is much smaller that the other. Consider a model problem $f(x_1, x_2) = \rho\,(x_1 - 0.45)^2 + (x_2 - 0.45)^2$ the scaling parameter is found both using solution of (6.16) with BFGS (black curve) and solution of Algorithm 6.2

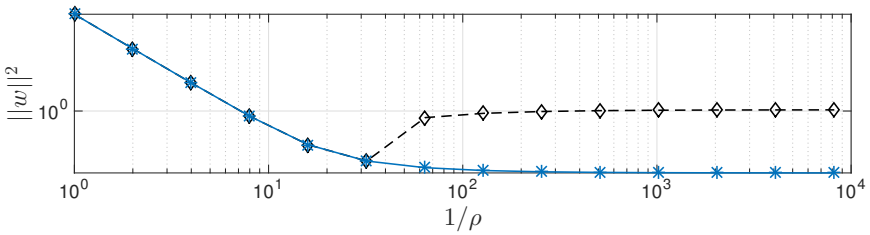


Figure 6.4: Vertical axis: the smoothness criteria $\|w\|^2$. Horizontal axis: the variation in x_1 direction respect to x_2 for a sample problem $f(x_1, x_2) = \rho\,(x_1 - 0.45)^2 + (x_2 - 0.45)^2$ both in loglog scale.

(blue curve). It is observed that even in a simple 2D problem using the same datapoints shown in Fig. 6.3, as one of the scaling parameters approaches zero, the direct SQP with BFGS solution is not able to provide a correct scaling parameter and converges to a local minimum. In contrast, the approach used in Algorithm 6.2, even for large ρ, does not deviate from the optimum path of the solutions, and converges to a better solution.

In Fig. 6.4 the black dashed line is the solution of (6.15) using direct approach and blue dashed-line is the solution of (6.15) using the iterative approach presented in Algorithm 6.2. In the plot $\frac{1}{\rho}$ is plotted. It is observed that for direct SQP with BFGS (black dashed line) converges to a local minimum of $\|w\|^2$ as $\rho \to 0$.

Algorithm 6.2 constantly initialize the solution of $Q_{\lambda_j}(\theta)$ for finding the $\theta_{\lambda_{k+1}}$ and enables Algorithm 6.2 to converge to the appropriate scaling solution.

6.5 Implementation of Δ-DOGS with both MAPS and PS on hydrofoil design

For validation purposes, Algorithm 6.1 was applied to the hydrofoil optimization problem considered in detail in [2], using both PS (6.8) and MAPS (6.11).

In [2], the shape of a racing catamaran's hydrofoil was characterized by 7 design parameters, and Δ-DOGS with PS was used to maximize the hydrofoil efficiency, defined as its lift/drag ratio, at a fixed working condition. During the optimization, two specific challenges were encountered: (*a*) Δ-DOGS apparently overexplored the function $f(x)$ near the boundary of feasibility, L, due in part to the fact that the objective function $f(x)$ itself had somewhat irregular behavior close to the boundary of L, and (*b*) the non-uniform dependence of the objective function $f(x)$ on the various design parameters x apparently resulted in overexploration of $f(x)$ in the vicinity of the optimizated solution. These challenges resulted in many apparently unnecessary function evaluations during the optimization. The issue described in (*a*) above was resolved well in [59, 73] by leveraging a grid or lattice in Δ-DOGS to help coordinate the search. In the following, we incorporate MAPS interpolation (6.11) in Δ-DOGS to resolve issue (*b*).

In our previous work on the hydrofoil optimization problem [2], we observed that the objective function $f(x)$ depended much more strongly on some design parameters than others. This nonuniform dependence the 7 adjustable parameters defining the hydrofoil shape (see Table 6.1 and Fig. 6.7) (note that, to ease the visualization, all

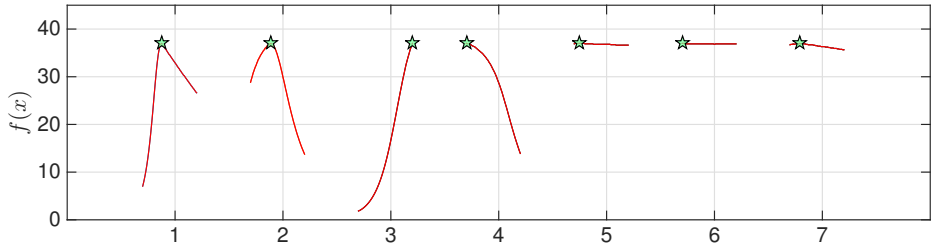


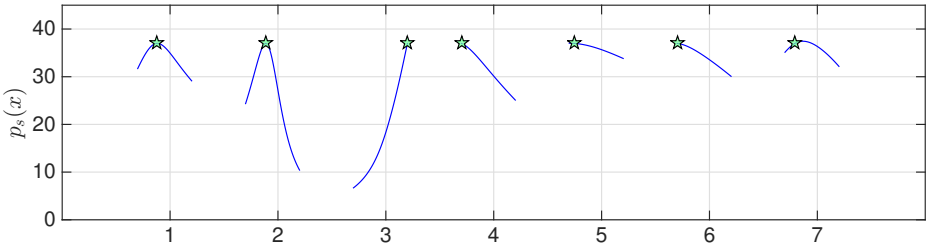
Figure 6.5: Nonuniform dependence of the lift/drag ratio of the hydrofoil on the 7 adjustable parameters defining its shape. Red line: variation of efficiency of the foil with 6 fixed and 1 moving parameters.

of the design parameters are normalized to lie between 0 and 1) in the vicinity of the optimum solution is shown in Fig. 6.5, where the The stars indicate the optimal values, x_ℓ^*, of each of the 7 parameters of the foil, and the red line indicates the variation in the efficiency of the foil as one of the parameters at a time is varied over its full range.

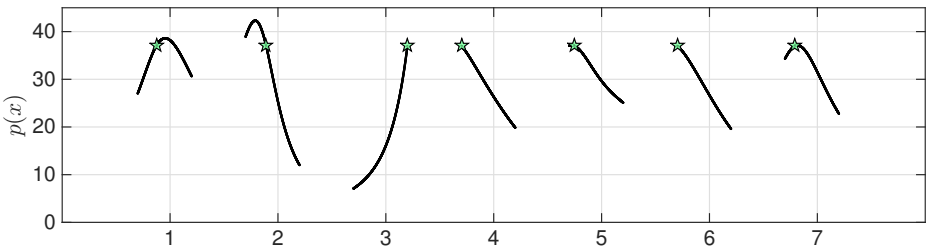
Simulation results (see Figure 6.8) indicate that, after about 30 function evaluations, the optimization algorithm approaches the optimal solution in the present problem, with a lift/drag ratio of about 36.

As mentioned previously (in particular, see Figure 6.3), spurious oscillations caused by nonuniform interpolants can be problematical in such optimization problems, significantly slowing convergence. We illustrate this issue by comparing PS (Figure 6.6, bottom) and MAPS (Figure 6.6, top) interpolations of the data given by the optimum solution x^* together with the first 30 datapoints generated by Algorithm 6.1. The PS interpolant is characterized by spurious peaks away from the optimal point, in both the x_1 and x_2 coordinate directions. These spurious peaks in the x_1 and x_2 coordinate directions are absent in the MAPS interpolant, which much more accurately captures the trends evident in the truth function itself (see Figure 6.5); this is remarkable, given that

the interpolation is based on only 31 datapoints in a practical 7-dimensional problem.



(a) MAPS (Eq. (6.11)).



(b) NPS (Eq. (6.8)).

Figure 6.6: MAPS v.s. NPS give the first 30 datapoints generated using Algorithm 6.1. Vertical: interpolant value for the efficiency of the foil with respect to adjustable parameters. Horizontal: normalized design parameters (Table 6.1).

We now summarize the design parameters used in this work, as suggested by [2], which represent a rectangular hydrofoil with an aspect ratio (AR) of 10 and a cross section of a NACA64$_1$ – 412 foil. The optimization is performed to minimize the drag of the foil subject to a design vertical and horizontal lift of $SC_z = 0.120$ and $SC_y = 0.066$.

Figure 6.7 shows the geometry of the foil, where z is the vertical coordinate, y is the horizontal cross-flow coordinate, x is the horizontal stream-wise coordinate, s is the curvilinear coordinate, and S is the planform area. Other parameters of the optimization govern the $y - z$ plan and the chord distribution along the curvilinear coordinate s. Both the shape of the foils' quarter-chord line and the chord distribution are represented using

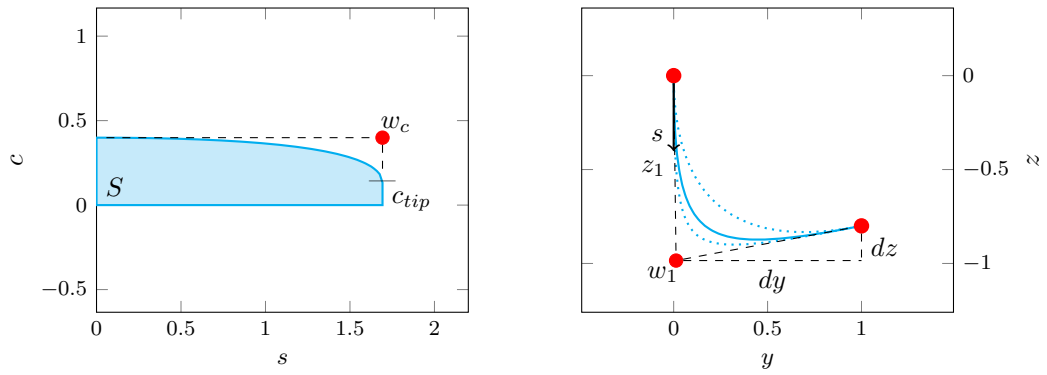


Figure 6.7: The optimization design parameters listed in Table 6.1 (see [2] for details of parametrization).

Table 6.1: Summary of the adjustable parameters used in the hydrofoil shape design problem.

Number (j)	Variable x_j	Description	lower bound	upper bound
1	S	planform surface	0.2	0.5
2	z_1	rational Bezier curve	0.5	1.5
3	dy	rational Bezier curve	0.5	1.5
4	dz	rational Bezier curve	-0.3	0.3
5	w_1	vertical plane weight	4.3	11
6	c_{tip}	tip chord length	0.05	0.5
7	w_c	weight spanwise	1.5	11

Bezier curves, where the w_i are the weights of the corresponding control points (for details, see [2]). In this manner, a realizable foil is characterized efficiently with only seven parameters subject to simple bound constraints, as listed in Table 6.1.

In this work, as suggested by and validated in [2], AVL has been used to calculate the lift/drag of a foil. AVL is an easy-to-use vortex-lattice-based software package for inviscid aerodynamic analysis problems of this sort. The estimate of the optimal objective function value, $y_0 = \max(C_L/C_D)$, can be obtained by classical aerodynamic analysis. The drag coefficient can be obtained as the sum of the viscous and inviscid (3D) drag components. For a foil with a specified aspect ratio of AR and an elliptic

spanwise load, it can be estimated as follows:

$$C_D = C_{D_v}(C_L) + \frac{C_L^2}{\pi AR}, \tag{6.20}$$

where $C_{D_v}(C_L)$ is the viscous drag coefficient, which can be determined for a given 2D foil section either via an experiment or a 2D computational model, such as XFoil [95]. Following the detailed analysis in [2], a value of $y_0 = 37$ is used in the present work.

6.5.1 Optimization results and comparison

The optimization process balances the contribution of the viscous drag, proportional to the foil surface, and the inviscid drag, proportional to the square of the lift surface. The objective function $f(x)$ used is the lift/drag ratio. We actually minimize $\log[1 + 1/f(x)]$, instead of maximizing the efficiency of the foil $f(x)$ itself, since Algorithm 6.1 is designed for minimization problems. Note that the plots provided in this paper show $f(x)$, for ease of interpretation.

By switching from Δ-DOGS with PS to Δ-DOGS(Λ) with MAPS interpolation, the numerical results in Figure 6.8 indicate a significant improvement in the objective function value, $f(x)$, after a fixed number of function evaluations. Implementing Δ-DOGS(Λ) with PS improves the coverage speed as compared with Δ-DOGS with PS. After only 33 function evaluations, Δ-DOGS(Λ) with PS finds a solution with a lift/drag ratio of 36; convergence to this level required 100 function evaluations using Δ-DOGS with PS. Moreover, using Δ-DOGS(Λ) with MAPS, a lift/drag ratio of 36 is achieved in

Table 6.2: Comparison between the results reported in [2] using Δ-DOGS with PS and Δ-DOGS(Λ) with MAPS for $dy < 1.50$.

variable	parameter	200 [Δ-DOGS]	40	80	200	269
x_1	S	0.305	0.305	0.305	0.297	0.303
x_2	z_1	0.89	0.900	0.875	0.862	0.881
x_3	dy	1.50	1.500	1.500	1.500	1.500
x_4	dz	-0.29	-0.300	-0.300	-0.300	-0.300
x_5	w_1	7.25	7.250	10.09	7.060	9.377
x_6	c_{tip}	0.21	0.163	0.129	0.100	0.050
x_7	w_c	2.58	2.896	2.896	2.814	3.220
$f(x)$	C_L/C_D	36.81	36.807	36.890	36.897	36.993

only 26 function evaluations. That is, Δ-DOGS(Λ) with MAPS has 74% improvement compared to the Δ-DOGS with PS, and a 21% enhancement compared to Δ-DOGS(Λ) with PS. Furthermore, after 55 function evaluations, Δ-DOGS(Λ) with MAPS found a solution with a lift/drag ratio of 36.89; on the other hand, [2] reported 160 function evaluations to reach a maximum lift/drag ratio of 36.81. Table 6.2 shows the optimized parameters of the hydrofoil design as computed in this paper, after various numbers of iterations, and in [2].

An important observations is that dy, the total length of the foil, reaches its maximum possible value in the optimized result. Intuition also suggests that, by increasing the foil aspect ratio, the foil efficiency will increase.

It can be seen in Figure 6.10 (top plot) that variables y_2 and z_2 do not vary after 30 function evaluations. This indicates it would be beneficial to project the optimization problem to a lower-dimensional parameter space, and to solve the optimization problem in that reduced parameter space. This idea, directly leveraging the MAPS formulation, will be investigated in future work.

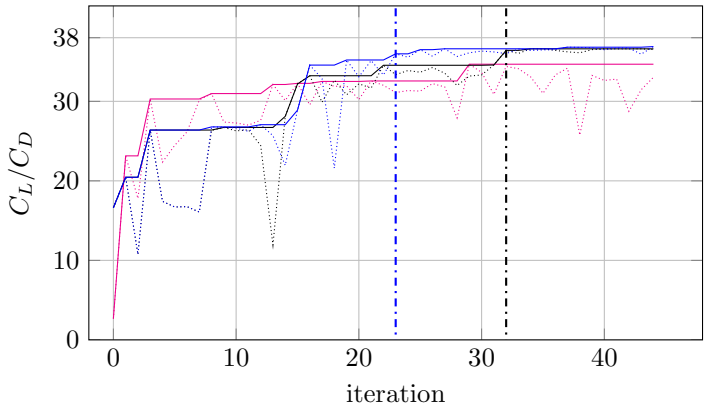
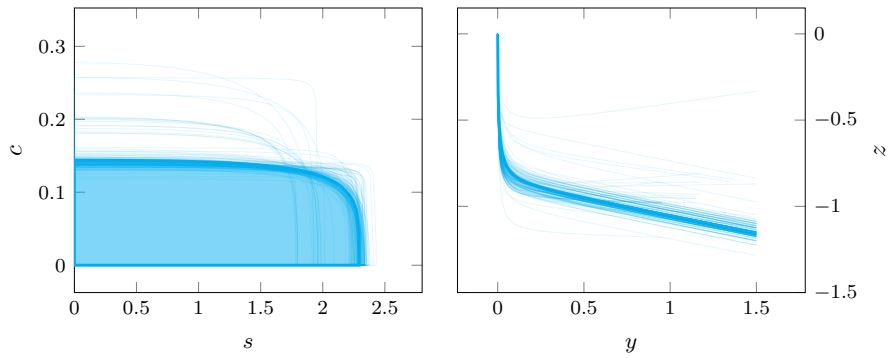


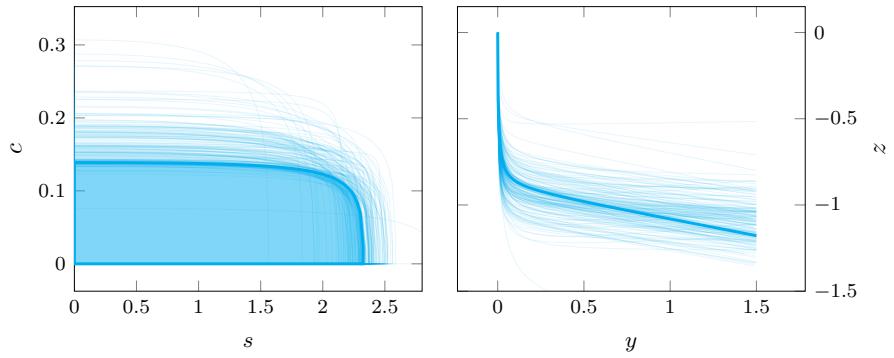
Figure 6.8: Solid lines show the best lift-drag ratio C_L/C_D at constant lift during the optimization: Blue curves illustrate the convergence of Δ-DOGS(Λ) w/ MAPS, Black curves w/ PS, and Magenta curves Δ-DOGS w/ PS [1], as reported in [2].

Convergence histories for the values of the optimized design parameters, using Δ-DOGS with PS [2] and Δ-DOGS(Λ) with MAPS, are shown in Figure 6.10. The designs achieved by both algorithms are illustrated in Figure 6.9. The final solutions obtained using both methods, illustrated in Figure 6.11, are quite similar. However, Figure 6.9a shows that the optimized design is found faster than in Figure 6.9b. These trends are also evident in Figure 6.10 and Figure 6.8.

Surrogate Management Framework (SMF) is one of the popular derivative-free methods that is a popular choice for shape optimization [29] [96]. Figure 6.12 illustrates the convergence of Δ-DOGS(Λ) w/ MAPS, Blue curves, and Surrogate Management Framework (SMF) w/ Kriging interpolation combined with Mesh Adaptive Search Method (MADS), Green curves. In this problem, Δ-DOGS(Λ) w/ MAPS outperforms SMF given the target value $y_0 = 37$ to both schemes.



(a) Δ-DOGS(Λ) with MAPS



(b) Δ-DOGS with PS [2]

Figure 6.9: The first 200 evaluated foil geometries in the optimization algorithms: *a*) Δ-DOGS(Λ) with MAPS. *b*) Δ-DOGS with NPS [2]. The optimized geometry is shown by a thick curve in both figures.

6.6 Conclusions

This paper presents a new interpolation strategy, multivariate adaptive polyharmonic spline (MAPS), for regularizing the interpolant used in response surface methods for derivative-free optimization. We have demonstrated that MAPS significantly accelerates the convergence rate of our own family of response surface methods, dubbed Δ-DOGS, when applied to practical design optimization problems of engineering interest.

MAPS is an interpolation strategy in the family of radial basis functions that

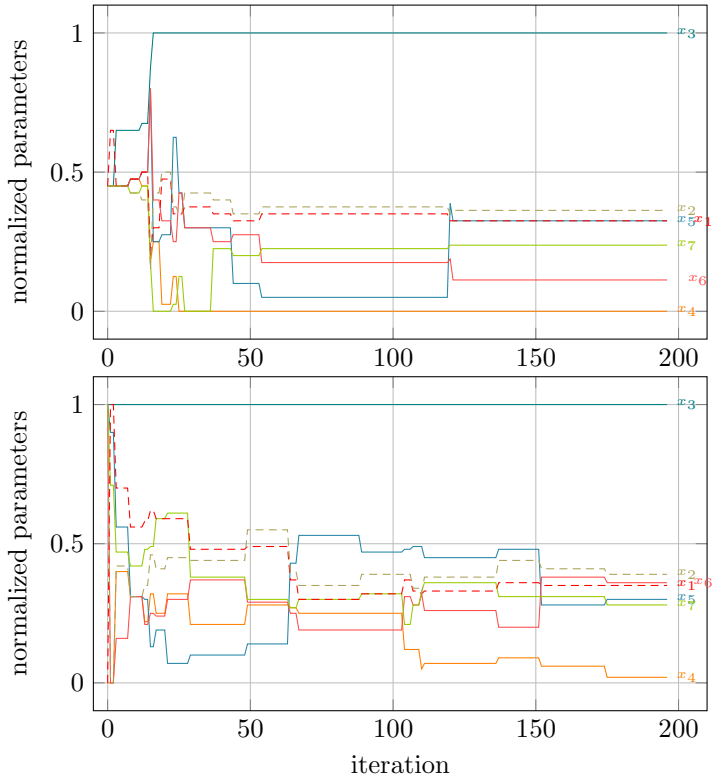


Figure 6.10: Convergence history of the optimal design parameter's values during the optimization. Top: Δ-DOGS(Λ) with MAPS. Bottom: Δ-DOGS [1] with PS. (as it is reported in [3, 2]). Dashed curves are the most dominant parameters, x_1 and x_2 in both plots.

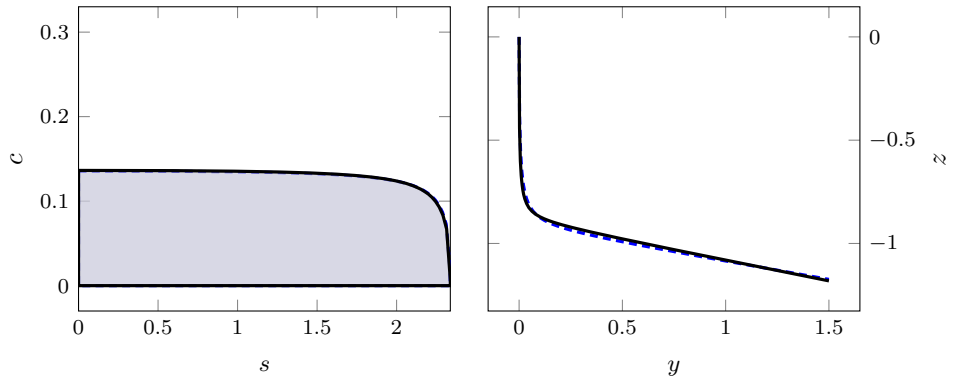


Figure 6.11: The optimum hydrofiol design using Δ-DOGS(Λ) with MAPS compared with the reported optimum foil in [2]. The dashed-blue is [2] and the solid line the optimum design reported in Table 6.2.

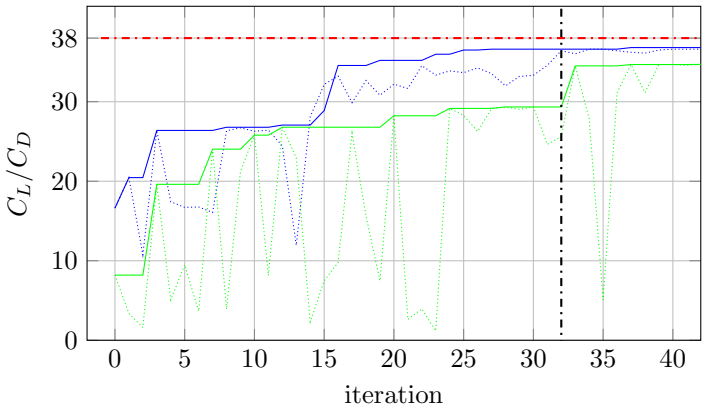


Figure 6.12: Solid lines show the best lift-drag ratio C_L/C_D at constant lift during the optimization: Blue curves illustrate the convergence of Δ-DOGS(Λ) w/ MAPS and Green curves SMF w/ Kriging interpolation combined with MADS.

rescales parameter space, based on the available data generated by an optimization algorithm, in order to reduce the spurious oscillations in the resulting interpolant. In the present work, MAPS is implemented in the Δ-DOGS family of optimization algorithms, and its performance is tested on a simulation-based shape design optimization problem to maximize the lift/drag ratio of a hydrofoil design. Results indicate fewer function evaluations are required following the new approach to achieve a given level of convergence. An improved method of coordinating the search with lattices is also introduced and shown to be effective.

A limitation of the interpolation strategy developed in this work is that its computational expense increases as the number of data points increases, since at each iteration it needs to solve an optimization problem to find the scaling parameters based on the available datapoints. However, in practice, the determination of these scaling parameters is found to be relatively inexpensive as compared with the function evaluations themselves, which are typically determined from expensive CFD simulations.

In future work, we will implement the present algorithm on additional test problems, and more computationally efficient implementations of MAPS will be developed. Also, the use of MAPS for the problem of dimension reduction in Δ-DOGS will be explored; that is, extending MAPS to aid in the exploration of $f(x)$ over a reduced number of parameters during intermediate steps of the optimization procedure.

Acknowledgements

Chapter 6 contains material that previously in part is published in: S. Alimohammadi, P. Beyhaghi, G. Meneghello, T. R. Bewley, (2017) Delaunay-based optimization in CFD leveraging multivariate adaptive polyharmonic splines (MAPS). *Proceedings of the 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, p. 129. The dissertation author was the primary researcher and author of this material.

Chapter 7

A Delaunay-based method for optimizing infinite time averages of numerical discretizations of ergodic systems: α-DOGSX

When something is important enough, you do it even if the odds are not in your favor. —Elon Musk

7.1 Introduction

The focus of this work is to develop a computational optimization technique that can be seamlessly applied to stochastic physical problems, for which the accurate evalu-

ation of the cost function involves a significant computational expense. In particular, we are interested in complex systems that require the time averaging of large-scale computations of partial differential equations (PDEs) during the design process. We aim to solve optimization problems whose objective function $f(x) : \Omega \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ has the following form:

$$f(x) = \lim_{T\to\infty} f_T(x), \quad f_T(x) = \frac{1}{T}\int_{t=0}^{T} y(t)dt, \tag{7.1}$$
$$dy(t)/dt = G(x,t),$$

where $G(x,t)$ is a nonlinear function of x and t. Note that x represents the vector of adjustable parameters in this dynamic system.

Moreover, we assume that the analytical expression for $G(x,t)$ is not available, and that we can only calculate $y(x,t\,h)$ numerically. In this way, the only available measurement for the objective function is

$$f(x,h,T) = \frac{1}{N}\sum_{i=1}^{N} y(x,i\,h), \quad N = \frac{T}{h}. \tag{7.2}$$

Note that $f(x,h,T)$ is a noisy measurement of $f(x)$, whose accuracy can be increased by decreasing h and increasing T; however, by doing so, the accuracy of these values is changed.

One approach for solving the above problem is via a derivative-based method using a finite difference estimate for the derivative. However, in order to capture the

behavior of the gradient of the objective function, a very accurate measurement is needed, which increases the computational cost of the optimization process. As a result, derivative-free methods offer a better choice for solving (7.1).

An important class of derivative-free algorithms is the family of Response Surface methods, which have been used to successfully solve problems, such as equation (7.2) [59, 97]. Among such models is the Delaunay-based optimization technique. Delaunay-based optimization is a generalizable family of practical, efficient, and provably convergent derivative-free algorithms designed for a range of nonconvex optimization problems with expensive function evaluations. Algorithms in this family are Response Surface methods that iteratively minimize metrics based on a surrogate model of existing datapoints, as well as on a synthetic model of the uncertainty of this surrogate.

One of the recently developed algorithms in this class, named α-DOGS, has been specifically designed to minimize the objective functions given by the infinite-time average of a statistically-stationary ergodic process; in such problems, any numerical or experimental approximation of the objective function is characterized by a sampling error, which may be reduced by additional sampling. The key idea behind the α-DOGS algorithm is that the number of samplings over different points in the feasible domain varies. This way, only a limited number of sampling is needed for points far from the solution, whereas in regions closer to the global minimum, a more extensive sampling is required for a precise quantification of the objective function, as convergence is approached. However, α-DOGS cannot handle the discretization error. Moreover, it is only designed for problems where the uncertainty is arise from the finite number of

sampling.

The existence of the discretization error is an important issue in many practical optimization problems generated by the numerical discretization of an original PDE problem of interest. Such challenges arise, for instance, in the minimization of the drag during the numerical approximation of a turbulent flow [88]. One approach to deal with discretization error is to use the same spatial grid for all measurements, thus reducing the problem to one grid. However, this approach is computationally expensive since it is not necessary to use a fine spatial grid throughout the entire optimization process.

In this paper, a new algorithm, dubbed α-DOGSX, is presented. It is an extended and modified version of α-DOGS, which efficiently automates the trade-off between (a) the additional sampling of the ergodic process, and (b) the refinement of the spatial discretization of the ODE (similar approach could be used for PDEs as well) [1]. Moreover, for a wide range of optimization problems having the same form as equation (7.1), a target value exists. In other words, we want to find control parameters x, so that $f(x) < f_0$, rather than actually minimizing $f(x)$. Therefore, in this paper, we also modify the α-DOGS in order to optimize the algorithm using the target value.

The structure of this paper is as follows: Section 7.2 briefly explains the essential elements of the α-DOGS for problems in which a target value exists. Section 7.3 presents the modified algorithm, which deals with the discretization error efficiently. Section 7.4 describes the performance of the algorithm when applied to a model problem

[1] The effect of domain size is also important. A comprehensive study on this issue is available in [98], regarding the box size effect in the Turbulence simulations. However, in most cases this error is relatively small and can be ignored in the computations.

based on the Lorenz equation. Finally, some conclusions are drawn in Section 7.5.

7.2 α-DOGS for problems with target value

In this section, we present the general framework of α-DOGS algorithm, which is designed to minimize objective functions of the form:

$$f(x) = \lim_{N\to\infty} \frac{1}{N} f_N(x), \quad f_N(x) = \sum_{i=1}^{N} f_i(x), \tag{7.3}$$

where $f_i(x)$ is a stationary and ergodic random process at each x.

Remark 24. *Estimating the value of the uncertainty associated with $f_N(x)$ is the classical uncertainty quantification (UQ) problem, studied in [99, 100, 101]. In this section, we will assume that this quantity is known, and it is denoted by $\sigma_N(x)$.*

The original algorithm [90] is designed to optimize the objective function of form (7.3) in general, but for a wide range of practical optimization problems (see [102, 103, 104]), we seek a point so that $f(x) \leq f_0$. In this section, a modified version of the algorithm described in [90] is presented. This version solves the optimization problems that have a target value f_0 more efficiently than before. This approach is similar to the EI (Expected Improvement), which is considered a major relevance criterion in global optimization [30, 75, 105].

Before presenting the algorithm, we briefly explain concepts originally defined in [1, 37, 58, 59, 60].

Definition 24. *Let S^k be a set of points in the feasible domain L, and $Y = \{f_{N_z^k}(z) | \, z \in S^k\}$ a set of measurements of $f(x)$ at these points. Moreover, let $p(x)$ be a regression that passed through these measurements, and f_0 a target value for $f(x)$. Then the continuous search function is defined as follows:*

$$s_c(x) = \begin{cases} \dfrac{p(x) - f_0}{e(x)} & \text{if } p(x) \geq f_0, \\ \\ p(x) - f_0 & \text{otherwise.} \end{cases} \tag{7.4}$$

In this framework, the regression $p(x)$ is a user-defined regression process, and $e(x)$ is the Delaunay-based uncertainty function defined in [37], which is a piece-wise quadratic function non-negative everywhere in L and equal to zero over S^k.

Definition 25. *Let us consider S^k a set of feasible points in L, and $Y = \{f_{N_z^k}(z) \mid z \in S^k\}$ a set of measurements of $f(x)$ at these points. Moreover, $\sigma_{N_z}(z)$ denotes the uncertainty of $f_{N_z^k}(z)$ and $p(x)$ is a regression for these measurements. Then the discrete search function, denoted $s_d(z)$ for each $z \in S^k$, is defined as follows. Note that this search function is defined only over S^k.*

$$s_d(z) = \begin{cases} \dfrac{\min\{p(z),\, 2\, f_{N_z}(z) - p(z)\} - f_0}{\sigma_{N_z}(z)} & \text{if } p(z) \geq f_0, \\ \\ p(z) - f_0 & \text{otherwise.} \end{cases} \tag{7.5}$$

In order to present the α-DOGS algorithm, we also need the concept of the Cartesian grid which has been initially defined in [59], and later extended to dense lattices in

[73].

Definition 26. *The Cartesian grid of level ℓ for the feasible domain $B = \{x|a \leq x \leq b\}$, denoted as B_ℓ, is:*

$$B_\ell = \left\{x|x = a + \frac{1}{N}(b-a) \otimes z, \quad z \in \{0, 1, \ldots, N\}\right\}, \text{ where } N = 2^\ell.$$

A quantizer of point $x \in B$ on B_ℓ is the point x_q, which has the minimum distance to x from the B_ℓ grid. Note that the quantizer of point x is not necessarily unique. The maximum discretization error is defined as follows:

$$\delta_\ell = max_{x \in B}||x - x_q||. \tag{7.6}$$

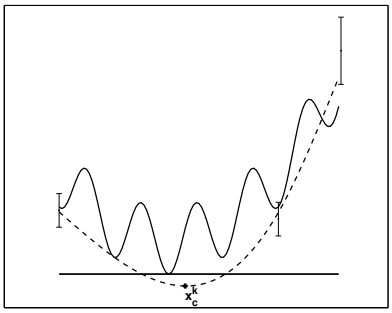
Illustration of the above concepts can be found in Fig. 2 of [59].

We can now present a modified α-DOGS algorithm, which allows us to find a point such that $f(x) \leq f_0$. The steps of this algorithm are summarized in Algorithm 7.1.

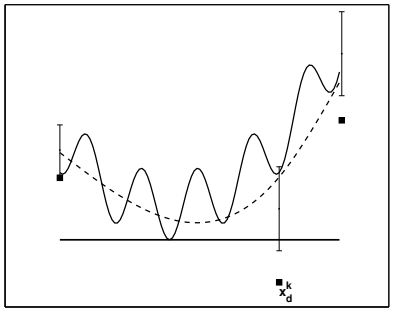
Remark 25. *Scaling the function evaluations and the domain of the parameters is required in order to have an efficient optimization algorithm. We will consider that $0 \leq x_i \leq 1$, and $0 \leq f(x) \leq 1$ after scaling.*

For each iteration of the algorithm, there are three possible situations:

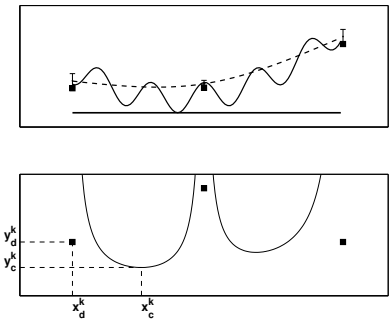
- A new point is added to the set of function evaluations. The iteration is called identifying sampling iteration (Fig. 7.1a and 7.1c).



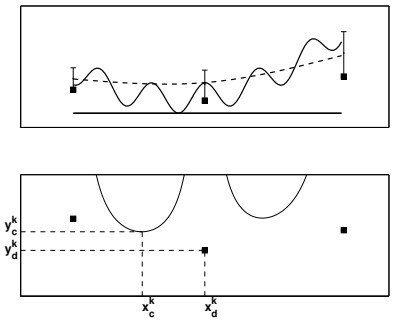
(a) identifying sampling iteration. Truth function $f(x)$ (dashed), target value f_0 (lower black line), regression model $p^k(x)$ (black), and $\sigma^k(x)$ (error bars).



(b) improving iteration. Truth function $f(x)$ (dashed), target value f_0 (lower black line), regression model $p^k(x)$ (black), and $\sigma^k(x)$ (error bars).



(c) identifying sampling iteration. Continuous search function $s_c^k(x)$, discrete search function $s_d^k(x)$, and their minimizers.



(d) improving iteration. Continuous search function $s_c^k(x)$, discrete search function $s_d^k(x)$, and their minimizers.

Figure 7.1: Elements of Algorithm 7.1.

- The averaging length at an existing point is improved. This iteration is called improving iteration (see Fig. 7.1b and 7.1d).

- The mesh level is increased. This iteration is called mesh refinery step.

Remark 26. *An important factor in Algorithm 7.1 is the construction of the regression $p^k(x)$. Generally, any well-behaved regression that is robust [90] can be used. However, in the simulation of this paper, we have used the efficient polyharmonic spline regression process (see [45, 59]).*

Remark 27. *Minimizing the search function $s_c^k(x)$ is an important sub-problem in Algorithm 7.1, as explained in [37].*

In addition to the regression process, there are a few parameters, which play a key role in Algorithm 7.1, and are summarized as follows:

a. β, which controls the amount of violation of the regression from the measurement at the available datapoints.

b. ℓ_0, which quantifies the mesh level at the first level. In the simulations of this paper, we have used $\ell_0 = 3$.

c. The initial averaging length N_0, and incrementing length N_1, which are problem-dependent parameters and control the minimum averaging length that should be considered between each two iterations of Algorithm 7.1.

One of the challenges we face in Algorithm 7.1 is determining the most promising point for the available measurements at the finite iteration k. This point is called the candidate and is determined as follows:

1. For $k = 0$, it is the point in the initial set whose measured value is minimized.

2. If step k is decreasing the mesh, then $\eta_k = q_k$.

3. If step k is improving iteration and w_k has the maximum averaging length over all points in S, then $\eta_k = w_k$.

4. If neither case 2 nor case 3 happens, $\eta_k = \eta_{k-1}$ for $k > 0$.

Algorithm 7.1 α-DOGS: find a point such that $f(x) \leq f_0$, where $f(x)$ is in the form given in equation (7.3)

0. Set $k = 0$ and initialize the algorithm with $\ell = \ell_0$ as the initial grid level, and a set of points S^k which includes some initial points on the initial grid of level ℓ_0. Afterwards, calculate an initial estimate with averaging length of N_0 at the initial points.

1. Calculate a robust regression [90] $p^k(x)$ over all available measurement such that

$$|p(z) - f_{N_z^k}(z)| \leq \beta \sigma_{N_z^k}(z), \quad \forall z \in S^k \tag{7.7}$$

   where N_z^k is the averaging length at point $z \in S^k$ at iteration k, and $\sigma_{N_z^k}(z)$ is the uncertainty associated with $f_{N_z^k}(z)$.

2. If a point $z \in S^k$ exists such that $\frac{p^k(z)-f_0}{\sigma_{N_z}(z)} \leq 1$, then increment the averaging length at z, N_z^k, by N_1; increment k and repeat from 1.

3. Find x_k and w_k, the minimizer of the continuous search function $s_c^k(x)$ and discrete search function $s_d^k(x)$.

4. If $s_c^k(x_k) \leq s_d^k(w_k)$, determine q_k a quantizer of x_k on the grid of level ℓ_k. Then, if $q_k \notin S^k$, add q_k to S^k and calculate an initial estimate of length N_0 at q_k; otherwise, refine the grid by incrementing ℓ_k by 1, and increment k.

5. If $s_c^k(x) > s_d^k(w_k)$, increase the averaging length at w_k by N_1.

6. Repeat steps 1 to 5 until the convergence is achieved.

In the next section, we analyze the convergence properties of Algorithm 7.1, and show that the value of the truth function at the candidate point ultimately achieve the target value, if the target value f_0 is achievable.

7.2.1 Convergence analysis of adaptive-K α-DOGS

The analysis presented in this section is similar to the one given in section 4 of [59]. Note that Algorithm 7.1 is a modified version of the Algorithm 1 described in

[90] for the range of problems where a target value is known; therefore, the analysis is similar.

In order to analyze the convergence properties of Algorithm 7.1, the following assumptions are made:

1. The truth function $f(x)$ and the regressions $p^k(x)$ are twice differentiable functions, and

$$-2\,\hat{K}I \leq \nabla^2 f(x) \leq 2\,\hat{K}I, \quad \forall x \in L, \tag{7.8}$$

$$-2\,\hat{K}I \leq \nabla^2 p^k(x) \leq 2\,\hat{K}I, \quad \forall x \in L. \tag{7.9}$$

2. The truth function $f(x)$ and the interpolating function $p^k(x)$ are Lipschitz with constant $\hat{L}$.

3. There is a constant γ such that for all measurements $f_N(x)$ at point x with uncertainty σ_N, then

$$|f_N(x) - f(x)| \leq \gamma\sigma_N(x). \tag{7.10}$$

This is a restrictive assumption for Algorithm 7.1. In practice, Algorithm 7.1 works well even if (7.10) is not verified, but in order to simplify the analysis, we make this additional assumption[2].

[2] In [90], a less restrictive assumption for the measurement process is imposed.

4. Defining $\sigma_N(x)$ as an uncertainty associated with the measurement $f_N(x)$, we have

$$0 < \sigma_N(x) \leq E(N), \tag{7.11}$$

$$\lim_{N\to\infty} E(N) = 0. \tag{7.12}$$

Note that $E(N)$ is considered to be a positive and monotonically decreasing function.

5. The target value f_0 is achievable.

Before analyzing the convergence of Algorithm 7.1, we present some preliminary lemmas.

Lemma 12. *At each iteration k of Algorithm 7.1, we have:*

$$\min\{s_c^k(x_k), s_d^k(w_k)\} \leq \max\{3\hat{K}, 2\gamma\} \tag{7.13}$$

Proof. In Lemma 4 of [90], it is shown that using Assumption 1, we have:

$$\min_{\{z\in S^k\}}\{2\, f(z) - p^k(z)\} + p^k(x) - 2f(x) - 3\,\hat{K}e^k(x) \leq 0, \tag{7.14}$$

for all $x \in L$. Note that $e^k(x)$ is the uncertainty function based on the Delaunay triangulation.

On the other hand, according to Assumption 4, for each point $z \in S^k$ with a

measurement $f_{N_z^k}(z)$ and an uncertainty $\sigma_{N_z}(z)$,

$$|f_{N_z^k}(z) - f(z)| \leq \gamma \sigma_{N_z}(z). \tag{7.15}$$

Thus, we have:

$$2\, f_{N_z^k}(z) - p^k(z) \leq 2\, f(z) - p^k(z) + 2\, \gamma \sigma_{N_z^k}(z) \tag{7.16}$$

where $f_{N_z^k}(z)$ is the measured value of z at iteration k with uncertainty $\sigma(z)$. Thus, using (7.14) and (7.16), we have:

$$\min_{\{z \in S^k\}} \{2\, f_{N_z^k}(z) - p^k(z) - 2\gamma \sigma_{N_z}(z)\}$$
$$+ p^k(x) - 2f(x) - 3\, \hat{K} e^k(x) \leq 0.$$

Now let us consider x^*, the global minimizer of $f(x)$ in L. Since f_0 is achievable (Assumption 4), then $f(x^*) \leq f_0$; therefore,

$$\min_{\{z \in S^k\}} \{2\, f_{N_z^k}(z) - p^k(z) - 2\gamma \sigma(z) - f_0\}$$
$$+ p^k(x^*) - f_0 - 3\, \hat{K} e^k(x^*) \leq 0,$$

$$\min\{\min_{\{z \in S^k\}} \{2\, f_{N_z^k}(z) - p^k(z) - 2\gamma \sigma_{N_z}(z) - f_0\},$$
$$p^k(x^*) - f_0 - 3\, \hat{K} e^k(x^*)\} \leq 0.$$

According to the construction of the discrete search function (7.5), to the fact that w_k is its minimizer, and to the fact that $\sigma_{N_z}(z) > 0$,

$$\min\{\sigma(z)\,(s_d^k(w_k) - 2\gamma),\, p^k(x^*) - f(x^*) - 3\,\hat{K}e^k(x^*)\} \leq 0. \tag{7.17}$$

However, if $s_d^k(w_k) - 2\gamma \leq 0$, then equation (7.13) is satisfied; thus, we will assume that $p^k(x^*) - f(x^*) - 3\,\hat{K}e^k(x^*) \leq 0$. Now, if $e^k(x^*) > 0$, then according to the construction of the discrete search function (7.5),

$$s_c^k(x^*) - 2\hat{K} \leq 0. \tag{7.18}$$

Furthermore, x_k is the minimizer of $s_c^k(x)$; thus, (7.13) is satisfied. The only case that is left is when $e^k(x^*) = 0$ and $p^k(x^*) - f_0 \leq 0$. Note that since $e^k(x^*) = 0$ by construction of the uncertainty function [37], with $x^* \in S^k$, and since $p^k(x^*) - f_0 \leq 0$, then $s_d^k(w_k) \leq s_d^k(x^*) \leq 0$. □

Lemma 13. *If iteration k of Algorithm 7.1 is a mesh refinery step, then*

$$f(y_{q_k}) - f_0 \leq (1 + \beta + \gamma)T, \tag{7.19}$$

$$T = \max\{E\,\delta_k^2, F\,\delta_k\},$$

$$E = 4\,\hat{K} + 2\,\gamma, \quad F = \hat{L}$$

where y_{q_k} and σ_k are the measurement and its uncertainty at point q_k. Note that since

this step is refines the mesh, q_k is in S^k. Moreover, δ_k is the quantization error at iteration k.

Proof. First, we will show that

$$y_{q_k} - f_0 \leq (1+\beta)T, \text{ and } \quad \sigma_{q_k} \leq T. \tag{7.20}$$

Since step k is mesh-decreasing, according to the construction of Algorithm 7.1, $R = s_c^k(x_k) \leq s_d^k(w_k)$. Then using Lemma 12, $R \leq \max\{3\hat{K}, 2\gamma\}$. If $R \leq 0$, then $p^k(x_k) \leq f_0$. Since $p^k(x_k)$ is Lipschitz, then

$$p^k(q_k) - f_0 \leq \hat{L}\delta_k,$$

$$\sigma_k \leq p^k(q_k) - f_0,$$

$$\sigma_k \leq \hat{L}\delta_k,$$

$$y_{q_k} - f_0 \leq \hat{L}(1+\beta)\delta_k.$$

Thus, equation (7.20) is satisfied. We will now consider the case where $R > 0$. In this case, since R is the minimizer of $s_c^k(x) = \frac{p^k(x)-f_0}{e^k(x)}$, then by construction x_k is also the minimizer of $G(x) = p^k(x) - Re^k(x)$. Using Assumption 1, equality $\nabla^2 e^k(x) = 2\,I$ (see [37]) and Lemma 3 in paper [90],

$$G(q_k) - G(x_k) \leq (\hat{K} + R)\delta_k^2 \leq (4\,\hat{K} + 2\,\gamma)\delta_k^2. \tag{7.21}$$

Moreover, $G(x_k) = f_0$ and since $q^k \in S^k$, then $e^k(q_k) = 0$.

$$p^k(q_k) - f_0 \leq (4\,\hat{K} + 2\,\gamma)\delta_k^2, \tag{7.22}$$

$$\sigma^k_{N_{q_k}} \leq p^k(q_k) - f_0, \tag{7.23}$$

$$f_{N^k_{q_k}}(q_k) - f_0 \leq (4\,\hat{K} + 2\,\gamma)(1+\beta)\delta_k^2. \tag{7.24}$$

Now using equation (7.20) and Assumption 3, (7.19) is satisfied. □

Lemma 14. *If iteration k of Algorithm 7.1 is an improving iteration, and thus increases the uncertainty of the maximum averaging length, then*

$$f(w_k) - f_0 \leq (\beta + \max\{3\hat{K}, 2\gamma\})E(N^k w_k), \tag{7.25}$$

where $N^k_{w_k}$ is the averaging length at w_k.

Proof. Since iteration k is improving, $s^k_d(w_k) \leq \max\{3\hat{K}, 2\,\gamma\,, 1\}$, using Lemma 12, then

$$\min\{p^k(w_k), 2\,\hat{f}(w_k) - p^k(w_k)\} - f_0 \leq \max\{3\hat{K}, 2\gamma\}\sigma^k_{w_k}(y_k). \tag{7.26}$$

Since the regression is robust,

$$f_{N^k_{w_k}}(w_k) - f_0 \leq (\beta + \max\{3\hat{K}, 2\gamma\})\sigma^k_{N_k}(w_k). \tag{7.27}$$

Now, according to Assumption 3, we have

$$f(w_k) - f_0 \leq (\gamma + \beta + \max\{3\hat{K}.2\gamma\})\sigma^k_{N_k}(w_k), \tag{7.28}$$

Thus, using assumption 5, equation (7.25) is shown. □

Theorem 8. *Let us consider η_k, the candidate point at iteration k. Then*

$$limsup_{k\to\infty} \quad f(\eta_k) - f_0 \leq 0. \tag{7.29}$$

Proof. According to the construction of the candidate point, its value is changed each time that we have either a mesh refinery step or an improving iteration, which maximizes the averaging length. Therefore, using Lemmas 13 and 14, equation (7.29) is satisfied if there is an infinite number of modifications in the value of η_k.

This theorem is shown by contradiction. Assuming that we have a finite number of mesh refinery steps, Algorithm 7.1 will obtain only a finite number of datapoints; thus, there must be an infinite number of improving iterations at a finite number of points. As a result, there is an infinite number of iterations that increase the averaging length at one of the available datapoints which has the maximum averaging length; this is a contradiction with our contradictory assumption; hence the proof of our theorem. □

7.3 Modified optimization algorithm to deal with discretization error: α-DOGSX

In this section, we modify Algorithm 7.1 to solve more general problems in the form of equation (7.1). The main difference between the objective function in equations (7.1) and (7.3) is that the convergence to $f(x)$ in (7.1) can be achieved only when h goes to zero. The steps of the modified algorithm α-DOGSX are the same as in Algorithm 7.1, but with a different improving iteration.

Recall that during each improving iteration of Algorithm 7.1, the averaging length at point w_k is simply incremented by N_1. In this modified algorithm, there are two possible options for improving the accuracy of the available measurement at point w_k:

a. Incrementing the value of the averaging length, T.

b. Decreasing the mesh size, h.

In order to develop an efficient trade-off between incrementing T and decreasing h, the following factors are important:

- The value of the uncertainty associated to the measurement $f(w_k, h_k, T_k)$ for $f(w_k)$, denoted by $\sigma(w_k, h_k, T_k)$.

- The required amount of improvement at point w_k.

- The computational cost of decreasing the measurement improvement.

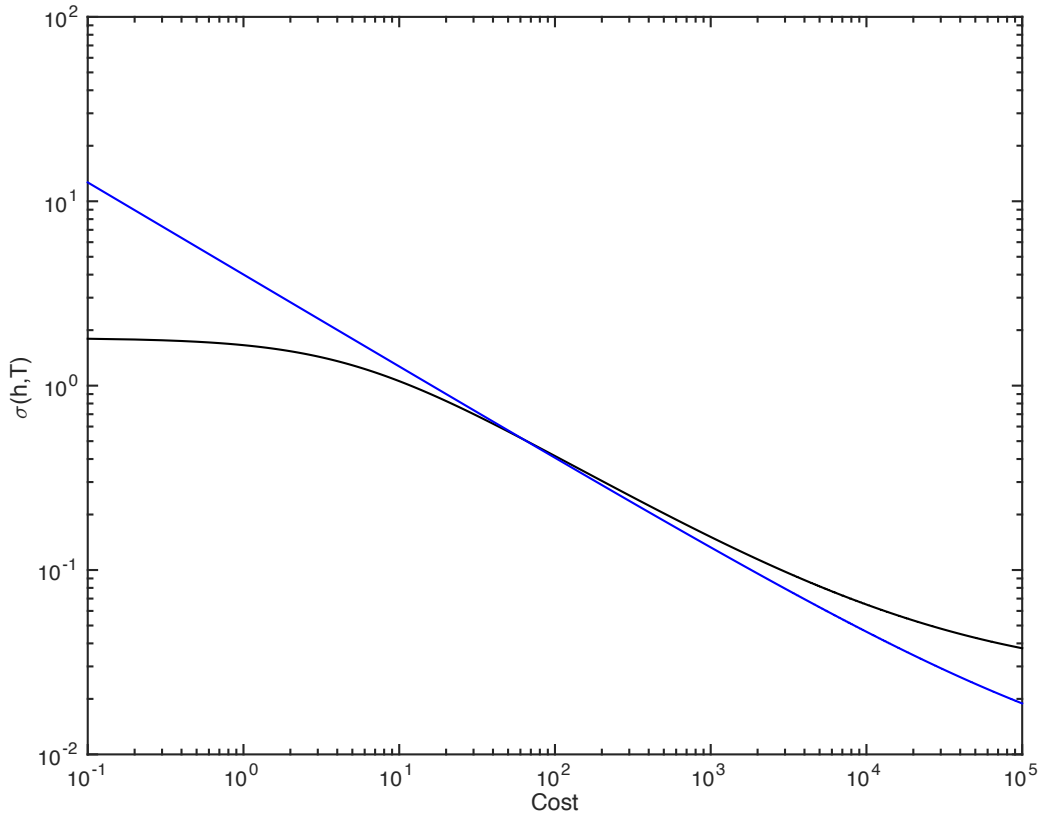


Figure 7.2: Effect of the required uncertainty on improvement. Black curve: trend of the uncertainty function at current h as a function of computational time. Blue curve: behaviour for a reduced h. For small values of σ_1^R, it is more efficient to increase the averaging length.

Remark 28. *The required amount of improvement plays a key role in our measurement improvement process. Fig. 7.2 illustrates the role of the required uncertainty. It is observed that in order to achieve a small amount of improvement, it is typically more efficient to increase the averaging length, T. However, for a greater improvement, the measurement should be made by modifying both h and T.*

We will now develop a procedure to estimate the required amount of reduction at point w_k. There are two possible cases for an improving step of the iteration k of

Algorithm 7.1:

1. Iteration k is improving since $s_d^k(w_k) \leq s_c^k(x_k)$. In this case, we expect to improve the uncertainty until $s_d^k(w_k) = s_c^k(x_k)$. Therefore, the required improvement at w_k, denoted by σ_k^R, is

$$\sigma_k^R = \frac{\min\{p^k(w_k), 2\, f_{N_w}(w_k) - p^k(w_k)\}}{s_c^k(x)}. \tag{7.30}$$

2. Iteration k is improving since a point $z \in S^k$ exists such that $p^k(z) - f_0 \leq \sigma^k(z)$. In this case, the required improvement at point z is obtained by

$$\sigma_k^R = \max\{p^k(z) - f_0, 0\}. \tag{7.31}$$

Note that expressions (7.30) and (7.31) estimate σ_k^R assuming that $p^k(x)$ is constant during the measurement improvement process. This assumption may not be true, especially if the reduction of the uncertainty is high. Thus, we will limit the reduction of the uncertainty to half of the current uncertainty.

Based on the available required improvement, the optimal procedure for the measurement improvement can be developed (Algorithm 7.2).

One of the challenges in Algorithm 7.2 is to solve (7.32). In general, this is an intricate problem, but for a wide range of practical problems [100, 103, 106], the value

Algorithm 7.2 In this algorithm, the process of measurement improvement at an improving step of Algorithm 7.1 is presented. In this algorithm, z is considered to be the point to be improved at iteration k. $T^k(z)$ and $h^k(z)$ are the current time averaging length and the mesh size of point z, which gives us a measurement of $f(z)$ with the uncertainty of $\sigma_k(z_k)$. $s_c^k(x)$ and $s_d^k(x)$ are the continuous and discrete search functions at the current iteration, whose minimizers are x_k and w_k respectively. The objective is to determine T_{k+1} and h_{k+1}.

1. If $z = w_k$, calculate σ_R^k using (7.30); otherwise, calculate via (7.31).
2. If $\sigma_R^k > \frac{\sigma_k(z_k)}{2}$ continue. Otherwise, consider $\sigma_R^k = \frac{\sigma_k(z_k)}{2}$.
3. For each $h_l = h_k\, 2^l$ for $l \in \{0, 1, \dots, \}$, calculate T_l to impose

$$\sigma(z, h_k\, 2^l, T_l) = \sigma_R^k, \tag{7.32}$$

   where $\sigma(z, h_l, T_l)$ is the uncertainty associated with the measurement at point z, with averaging length T_l and mesh size of h_l.

4. For each $l \in \{0, 1, \dots, \}$, calculate the cost associated with each required measurement as follows:

$$Cost_l = \begin{cases} \frac{T_l}{h_k\, 2^l} & \text{if } l = 0, \\ \\ \frac{T_l - T_k}{h_k} & \text{otherwise.} \end{cases} \tag{7.33}$$

   Note that in the above expression, we estimate the cost of the measurement process by the number of time marching required.

5. Calculate l_{opt} as the minimizer of $cost_l$, and take $h_{k+1} = h_l\, 2^l$, and $T_{k+1} = T_l$.

of $\sigma(z, h, T_l)$ has the following expression:

$$\sigma(z, h, T) = C_0(z)\, h^p + \frac{\sigma_0(z)}{\sqrt{T}}. \tag{7.34}$$

Using equation (7.34) as a model for the value of the uncertainty, the constant $C_0(z)$ and $\sigma_0(z)$ are determined empirically, based on the available measurement [100].

7.4 Implementation of α-DOGSX on a model problem

In this section, we apply the optimization algorithm we presented in the previous section on a model problem based on the Lorenz system.

The Lorenz system is a strange attractor that arises in a system of equations describing the 2-dimensional flow of a fluid of uniform depth, with an imposed vertical temperature difference. The chaotic behavior of a simplified 3-dimensional system of this problem, known as the Lorenz equations [104], is given below:

$$\frac{d}{dt}X = s\,(Y - X) \tag{7.35}$$

$$\frac{d}{dt}Y = -XZ + r\,X - Y \tag{7.36}$$

$$\frac{d}{dt}Z = XY - b\,Z \tag{7.37}$$

Let us consider an estimation problem in which the values of various moments of the chaotic attractor (see, e.g., Table 3 in [104]) at the nominal values of the parameters are taken as the target. The goal is to search over the parameter space (r, b, s) in an attempt

to determine the parameter values $(28, 8/3, 10)$.

Solving Lorenz system using α-DOGSX

We consider different moments as objective function, J, for optimization:

$$J(r, b, s) = \sum_{i=1}^{K} (f_i - f_{i,\text{target}})^2 \tag{7.38}$$

where f_i represent different moments, such as the expected value of Z, $E[Z]$. The reported target value, f_0, using RK4 for time marching of (7.35) for $E[Z]$ is 23.57 [100] [104]. In this problem, we take r as the design parameter.

In the Lorenz system, the values of r, b, and s are positive. The Lorenz system becomes chaotic if [104]

$$r > s\frac{s + b + 3}{s - b - 1}. \tag{7.39}$$

Moreover, r is related to the finite time averaging quantity of Z in (7.35), denoted as $\bar{Z}$. Using this relationship, the upper bound for r can be estimated by:

$$\bar{Z}^i \leq (r - 1)^i, \quad i = 1, 2, 3. \tag{7.40}$$

Imposing (7.40) and (7.39), we choose $24 \leq r \leq 30$, $b = 8/3$ and $s = 10$. It has been reported in [100] that the discretization error of $\bar{Z}$, ε_h, can be modeled as $C_0 h^p$ for the Lorenz system where $p = 3$ and $C_0 = 0.8$ approximately.

Table 7.1 and Fig. 7.3 show the points generated during the optimization algo-

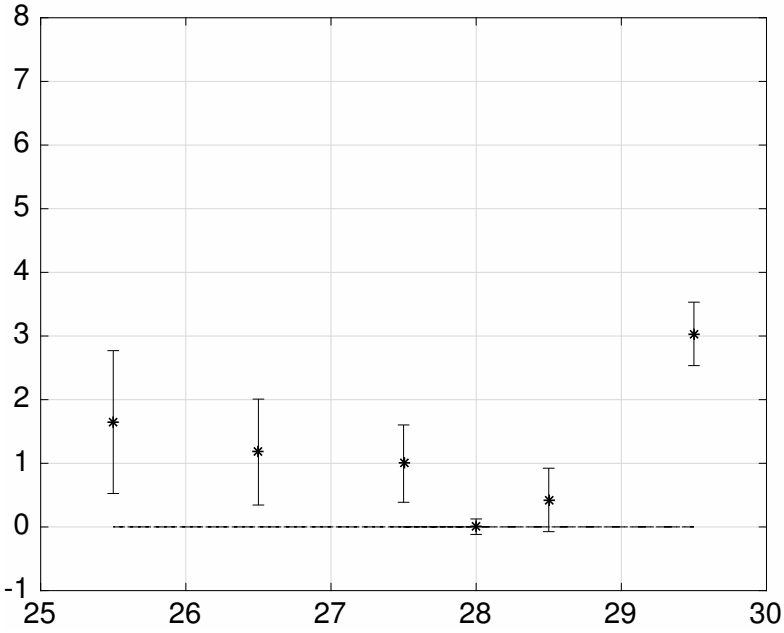


Figure 7.3: Illustration of the points generated by α-DOGSX on Lorenz problem.

rithm. The initialization is performed with an initial time averaging of $T_0 = 50$, and the solver grid size is $h_0 = 2.5 \times 10^{-3}$ for α-DOGS and $h_0 = 0.05$ for α-DOGSX.

Table 7.1: The points generated by applying the optimization algorithm to the Lorenz system.

r	27.5	28.5	29.5	25.5	26.5	28.
J	0.9	0.42	3.0	1.65	1.17	10^{-3}
T	50	360	280	210	300	570
h	0.05	2×10^{-4}	1×10^{-3}	3×10^{-3}	8×10^{-3}	5×10^{-4}
Cost	10^3	10^6	10^5	10^4	10^5	10^7

7.5 Conclusions

In this paper, we developed a new algorithm, dubbed α-DOGSX, for solving optimization problems, whose functions are the infinite time-averaged statistics of a continuous stationary process without discretization error. The method we presented is

based on the Delaunay-based optimization algorithm which is developed in [90].

We used the finite-time averaging value, which is an estimate for the truth objective value at each given resolution of the mesh grid. The novelty of our algorithm is that the different averaging lengths and mesh grid resolutions are used at different sets of design parameters. The flexibility in the calculation of function evaluations that are less accurate far from the solution allows us to get an efficient algorithm for solving problems whose function values are given.

Our algorithm has a tuning parameter α, which plays a key role in the optimization process. It is observed that as α decreases, we use more data points for the convergence, but the total amount of averaging decreases. Note, that this flexibility is useful since for some applications (e.g., turbulence simulations), it is more efficient to use fewer datapoints, as the examination of each data point is costly (because of the complexity in setting up a measurement process at each point). Note also that in the limit where the initialization cost is too expensive, it is better to use the same averaging length for all data points.

Another important issue which has to be considered is the role of the desired accuracy in our algorithm. It is observed that as the desired accuracy increases, the efficiency of the optimization process increases as well. In other words, in the limit where the desired accuracy goes to 0, the computational cost of the optimization process converges to the computational cost of the most expensive measurement.

Furthermore, since the stated problem (7.1) is stochastic, the performance of the presented algorithms are also stochastic. In other words, based on the measured values

that are derived as the algorithm proceeds, the convergence speed changes; however, the main advantage of this algorithm is that the convergence is guaranteed under some conditions.

Although this new method works well for the test functions we presented in this paper, it cannot be practically used for high-dimensional problems yet, due to the exponential growth of the number of simplices with the dimension. This is an important limitation of our optimization algorithm.

In our future work, we intend to implement our algorithm for some practical optimization problems in turbulence simulations.

Acknowledgements

Chapter 7 is currently being prepared for submission for publication. P. Beyhaghi, S. R. Alimo, T. R. Bewley, “A Delaunay-based method for optimizing infinite time averages of numerical discretizations of ergodic systems”. The dissertation author was the primary researcher and author of this material.

Chapter 8

Conclusions and future work

The thesis presents significant extensions of the new family of surrogate-based derivative-free optimization schemes, dubbed Δ-DOGS. The idea unifying this efficient, and under the appropriate assumptions provably-globally-convergent, family of schemes is the minimization of a search function that linearly combines: (1) a computationally inexpensive "surrogate" (an interpolation, or in some cases a regression, of recent function evaluations) to summarize the trends evident in the data available thus far; with (2) a synthetic piecewise-quadratic "uncertainty function" that is built on the framework of a Delaunay triangulation of existing datapoints, and that characterizes the reliability of the surrogate by quantifying the distance of any given point in parameter space to the nearest function evaluations. In this thesis, five optimization algorithms are introduced.

The first algorithm called Δ-DOGS(Ω), a new Delaunay-based derivative-free optimization method. Δ-DOGS(Ω) is designed specifically for the optimization of computationally expensive nonconvex functions within a nonconvex (possibly lower dimen-

sional, or even disconnected) feasible domain, which is itself defined by computationally expensive constraint functions that are explored as the iterations proceed. Two main variants of Δ-DOGS(Ω) have been presented. Here, a search function defined by comparing simple functions of the uncertainty model and the interpolants (of both the objective and the constraint functions) is minimized within the rectangular search domain at each iteration. Next, a new objective and constraint function computations are performed at the optimized point, thereby refining the surrogate models of the objective and constraint functions at each iteration until convergence is achieved. Convergence to the feasible global minimum is proved mathematically under reasonable technical conditions on the smoothness of the objective and constraint functions.

This optimization algorithm has two main limitations. First, Δ-DOGS(Ω) over-explores the boundary of search domain, which increases the number of expensive function evaluations. Second, the initialization process requires function evaluations at all vertices of the search domain.

The second algorithm, Δ-DOGS(Ω_Z), modifies Δ-DOGS(Ω) to overcome the above two issues. The modified algorithm reduces the number of function evaluations on the boundary of the search domain, and at the initialization step. This is done by introducing a new search function, as well as splitting the datapoints into evaluated and unevaluated points, which improve the behavior of uncertainty near the boundary of search domain.

The third algorithm is a hybrid approach to leverage the gradient information when available. We propose a trust region method using the Voronoi cell around the

best available point. The hybrid algorithm blends derivative-free global exploration with derivative-based local refinement, inherits the proof of global convergence (under the appropriate assumptions) of the Δ-DOGS algorithm. In our numerical experiments, the algorithm is found to handle well nonconvex functions with many local minima, and to scale better with dimension than purely derivative-free global optimization approaches.

The fourth algorithm is called Δ-DOGS(Λ). The algorithm modifies the search domain to handle a general linear constraint domain. In previous chapters, a Cartesian grid was used to regularize the datapoints during the optimization algorithm. The extension to general linear constraints and other dense lattices are highlighted in Δ-DOGS(Λ), which outperforms the original Δ-DOGS algorithms [3]. Thus, Δ-DOGS(Λ) offers much greater potential for high-level applications that require linear search domains that the previous versions of Δ-DOGS may not have been able to handle as efficiently or effectively.

The fifth algorithm is called α-DOGSX, which is a variant designed to simultaneously increase the statistical sampling and refine the numerical approximation of the function evaluations as convergence is approached. This novel algorithm is highly application-driven, and addresses most of the practical challenges in the optimization of computer-aided shape designs.

In addition to the five new algorithms, we developed a new interpolation strategy, MAPS, as a feature of Δ-DOGS. In practical applications where the objective function is non-uniformly dependent on parameters, off-the-shelf interpolation strategies may be insufficient or ineffective in predicting the trends of the objective function. When coupled

with Δ-DOGS, MAPS accelerates the local convergence of algorithms to more accurately provide the trend. The superior performance of MAPS compared to benchmark strategies was verified in generating the most efficient hydrofoil design.

To demonstrate practical applicability of the algorithms, the following optimization problems were explored: (A) the design of low-storage implicit/explicit RungeKutta (IMEXRK) schemes for high performance computing (HPC) problems, such as the numerical simulation of turbulence; (B) the shape optimization of the hydrofoil to achieve the maximum efficiency. We are currently optimizing the wall texture for fully developed turbulent flow drag reduction using the algorithms.

Clearly, the presented algorithms are powerful tools, but they do have their limitations, and the next step will be to address the challenges and limitations. The main limitation of the response surface methods developed in the present thesis, and the related optimization algorithms in [1, 37], is the memory requirements related to the Delaunay triangulation algorithms. The "curse of dimensionality", is the main limitation in this family of algorithms due to the exponential growth of a number of simplices as the dimensions increases. As a result, constructing Delaunay triangulations for problems of more than ten dimensions is computationally intractable. Using some dimension reduction methods based on the surrogate model of available existing points is an attractive research to be considered.

Overall, the test problems considered in this thesis effectively illustrate the essential features of the new data-driven optimization algorithms. In the future, we will test these data-driven optimization algorithms on application-based problems to improve

the engineering designs.

Bibliography

[1] P. Beyhaghi, T. Bewley, Delaunay-based derivative-free optimization via global surrogates, part 2: convex constraints, Journal of Global Optimization (2016) under review.

[2] G. Meneghello, P. Beyhaghi, T. R. Bewley, Simulation-based optimization of the hydrofoil of a flying catamaran.

[3] P. Beyhaghi, Delaunay-based global optimization algorithms and their applications.

[4] D. Cavaglieri, T. Bewley, Low-storage implicit/explicit Runge-Kutta schemes for the simulation of stiff high-dimensional ODE systems, J. Comput. Phys. 286 (2015) 172–193.

[5] A. L. Marsden, M. Wang, J. E. Dennis, Jr., P. Moin, Optimal Aeroacoustic Shape Design Using the Surrogate Management Framework, Optimization and Engineering 5 (2) (2004) 235–262.

[6] B. F. Pussoli, L. W. D. Jader R. Barbosa, M. Kaviany, Optimization of peripheral finned-tube evaporators using entropy generation minimization, International Journal of Heat and Mass Transfer 55 (25-26) (2012) 7838–7846. doi:10.1016/j.ijheatmasstransfer.2012.08.021.

[7] M. E. Moghadam, F. Migliavacca, I. E. Vignon-Clementel, T.-Y. Hsia, A. L. Marsden, Optimization of shunt placement for the norwood surgery using multi-domain modeling, Journal of Biomechanical Engineering 134 (5) (2012) 051002.

[8] C. C. Long, A. L. Marsden, Y. Bazilevs, Shape optimization of pulsatile ventricular assist devices using FSI to minimize thrombotic risk, Computational Mechanics 54 (4) (2014) 921–932. doi:10.1007/s00466-013-0967-z.

[9] S. Alimohammadi, D. He, Multi-stage algorithm for uncertainty analysis of solar power forecasting, in: Power and Energy Society General Meeting (PESGM), 2016, IEEE, 2016, pp. 1–5.

[10] R. Madani, M. Ashraphijuo, J. Lavaei, Promises of conic relaxation for contingency-constrained optimal power flow problem, IEEE Transactions on Power Systems 31 (2) (2016) 1297–1307.

[11] P. E. Gill, W. Murray, M. A. Saunders, M. H. Wright, A schur-complement method for sparse quadratic programming., Tech. rep., DTIC Document (1987).

[12] P. E. Gill, W. Murray, M. A. Saunders, SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization, SIAM Journal on Optimization 12 (4) (2002) 979–1006.

[13] R. Fletcher, S. Leyffer, P. L. Toint, On the Global Convergence of a Filter–SQP Algorithm, SIAM Journal on Optimization 13 (1) (2002) 44–59. doi:10.1137/S105262340038081X.
URL http://epubs.siam.org/doi/abs/10.1137/S105262340038081X

[14] A. Forsgren, P. E. Gill, M. H. Wright, Interior Methods for Nonlinear Optimization, Vol. 44, SIAM, 2002. doi:10.1137/S0036144502414942.

[15] D. P. Bertsekas, On penalty and multiplier methods for constrained minimization, SIAM Journal on Control and Optimization 14 (2) (1976) 216–235.

[16] B. Kort, D. Bertsekas, A new penalty function method for constrained minimization, in: Decision and Control, 1972 and 11th Symposium on Adaptive Processes. Proceedings of the 1972 IEEE Conference on, IEEE, 1972, pp. 162–166.

[17] B. W. Kort, D. P. Bertsekas, Combined primal-dual and penalty methods for convex programming, SIAM Journal on Control and Optimization 14 (2) (1976) 268–294.

[18] A. Forsgren, P. E. Gill, Primal-Dual Interior Methods for Nonconvex Nonlinear Programming, SIAM Journal on Optimization 8 (4) (1998) 1132–1152. doi:10.1137/S1052623496305560.

[19] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, Journal of Global optimization 13 (4) (1998) 455–492.

[20] V. Torczon, On the convergence of pattern search algorithms, SIAM Journal on optimization 7 (1) (1997) 1–25.

[21] R. M. Lewis, V. Torczon, Pattern search algorithms for bound constrained minimization, SIAM Journal on Optimization 9 (4) (1999) 1082–1099.

[22] R. M. Lewis, V. Torczon, Pattern search methods for linearly constrained minimization, SIAM Journal on Optimization 10 (3) (2000) 917–941.

[23] R. M. Lewis, V. Torczon, A globally convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds, SIAM Journal on Optimization 12 (4) (2002) 1075–1089.

[24] M. A. Abramson, C. Audet, J. E. Dennis, S. L. Digabel, OrthoMADS: A Deterministic MADS Instance with Orthogonal Directions (2009).

[25] C. Audet, J. E. Dennis, A Pattern Search Filter Method for Nonlinear Programming without Derivatives, SIAM Journal on Optimization 14 (4) (2004) 980–1010.

[26] C. Audet, J. E. Dennis, Mesh Adaptive Direct Search Algorithms for Constrained Optimization, SIAM Journal on Optimization 17 (1) (2006) 188–217.

[27] C. Audet, J. E. Dennis, A Progressive Barrier for Derivative-Free Nonlinear Programming, SIAM Journal on Optimization 20 (1) (2009) 445–472.

[28] P. Belitz, T. Bewley, New horizons in sphere-packing theory, part ii: lattice-based derivative-free optimization via global surrogates, Journal of Global Optimization 56 (1) (2013) 61–91.

[29] A. J. Booker, J. Dennis J.E., P. D. Frank, D. B. Serafini, V. Torczon, M. W. Trosset, A rigorous framework for optimization of expensive functions by surrogates, Structural and Multidisciplinary Optimization 17 (1) (1999) 1–13.

[30] D. R. Jones, A taxonomy of global optimization methods based on response surfaces, Journal of global optimization 21 (4) (2001) 345–383.

[31] R. B. Gramacy, G. A. Gray, S. Le Digabel, H. K. Lee, P. Ranjan, G. Wells, S. M. Wild, Modeling an augmented lagrangian for blackbox constrained optimization, Technometrics 58 (1) (2016) 1–11.

[32] M. Schonlau, W. J. Welch, D. R. Jones, A data-analytic approach to bayesian global optimization, in: Department of Statistics and Actuarial Science and The Institute for Improvement in Quality and Productivity, 1997 ASA conference, 1997.

[33] S. M. Wild, R. G. Regis, C. A. Shoemaker, Orbit: Optimization by radial basis function interpolation in trust-regions, SIAM Journal on Scientific Computing 30 (6) (2008) 3197–3219.

[34] D. G. Krige, A statistical approach to some mine valuation and allied problems on the witwatersrand, Ph.D. thesis (1951).

[35] J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn, Design and Analysis of Computer Experiments, Statistical Science 4 (4) (1989) 409–423. doi:10.1214/ss/1177012413.

[36] C. E. Rasmussen, Gaussian processes for machine learning.

[37] P. Beyhaghi, D. Cavaglieri, T. Bewley, Delaunay-based derivative-free optimization via global surrogates, part i: linear constraints, Journal of Global Optimization 1–52.

[38] S. Hornus, J.-D. Boissonnat, An efficient implementation of delaunay triangulations in medium dimensions.

[39] A. Wächter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Mathematical programming 106 (1) (2006) 25–57.

[40] D. F. Watson, Computing the n-dimensional delaunay tessellation with application to voronoi polytopes, The computer journal 24 (2) (1981) 167–172.

[41] K. L. Hoffman, A method for globally minimizing concave functions over convex sets, Mathematical Programming 20 (1) (1981) 22–32.

[42] C. T. Lawrence, J. L. Zhou, A. L. Tits, User's guide for cfsqp version 2.0: Ac code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints.

[43] E. Pee, J. O. Royset, On solving large-scale finite minimax problems using exponential smoothing, Journal of Optimization Theory and Applications 148 (2) (2011) 390–421.

[44] E. Polak, J. O. Royset, R. S. Womersley, Algorithms with adaptive smoothing for finite minimax problems, Journal of Optimization Theory and Applications 119 (3) (2003) 459–484. doi:10.1023/B:JOTA.0000006685.60019.3e.

[45] G. Wahba, Spline models for observational data, Vol. 59, Siam, 1990.

[46] P. A. Simionescu, Computer-aided Graphing and Simulation Tools for AutoCAD Users, Vol. 32, CRC Press, 2014.

[47] S. Le Digabel, Algorithm 909: Nomad: Nonlinear optimization with the mads algorithm, ACM Transactions on Mathematical Software (TOMS) 37 (4) (2011) 44.

[48] M. D. McKay, R. J. Beckman, W. J. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 42 (1) (2000) 55–61.

[49] A. L. Marsden, J. A. Feinstein, C. A. Taylor, A computational framework for derivative-free optimization of cardiovascular geometries, Computer Methods in Applied Mechanics and Engineering 197 (21) (2008) 1890–1905.

[50] A. B. Ramachandra, S. Sankaran, J. D. Humphrey, A. L. Marsden, Computational simulation of the adaptive capacity of vein grafts in response to increased pressure, Journal of biomechanical engineering 137 (3) (2015) 031009.

[51] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier-stokes Equations, J. Comput. Phys. 59 (1985) 308–323.

[52] J. Kim, P. Moin, B. Moser, Turbulence statistics in fully developed channel flow at low Reynolds number, J. Fluid Mech. 177 (1987) 133–166.

[53] H. Le, P. Moin, An improvement of fractional step methods for the incompressible Navier-Stokes equations, J. Comput. Phys. 92 (1991) 369–379.

[54] A. Wray, Minimal-storage time advancement schemes for spectral methods.

[55] P. Spalart, R. Moser, M. Rogers, Spectral methods for the Navier-Stokes equations with one infinite and two periodic directions, J. Comput. Phys. 96.2 (1991) 297–324.

[56] J. Chifman, L. Kubatko, Quartet inference from snp data under the coalescent model, Bioinformatics 30 (23) (2014) 3317–3324.

[57] D. Cavaglieri, P. Beyhaghi, T. Bewley, Low-storage imex runge-kutta schemes for the simulation of navier-stokes systems, in: 21st AIAA computational fluid dynamics conference, San Diego, CA, 2013.

[58] P. Beyhaghi, D. Cavaglieri, T. Bewley, Delaunay-based derivative-free optimization via global surrogates, part i: linear constraints, Journal of Global Optimization 66 (3) (2016) 331–382.

[59] P. Beyhaghi, T. Bewley, Delaunay-based via lattice-based optimization algorithm, Journal of Global Optimization (2016) under review.

[60] S. R. Alimo, D. Cavaglieri, P. Beyhaghi, T. R. Bewley, Design of imexrk time integration schemes via delaunay-based derivative-free optimization with nonconvex constraints and grid-based acceleration., Journal of Global Optimization, Under preparation.

[61] J. Butcher, Numerical methods for ordinary differential equations, Wiley, 2008.

[62] J. Dormand, P. Prince, A family of embedded Runge-Kutta formulae, J. Comput. & Appl. Math. 6.1 (1980) 19–26.

[63] C. Kennedy, M. Carpenter, R. Lewis, Additive Runge-Kutta schemes for convection-diffusion-reaction equations, Appl. Num. Math. 44 (2003) 139–181.

[64] M. Calvo, J. de Frutos, J. Novo, Linearly implicit Runge-Kutta methods for advection-reaction-diffusion equations, Appl. Num. Math. 37 (4) (2001) 535–549.

[65] C. Kennedy, M. Carpenter, Additive Runge-Kutta schemes for convection-diffusion-reaction equations, Tech. rep., NASA Tech. Rep. (2001).

[66] A. Torn, A. Zilinskas, Global optimization, Springer-Verlag New York, Inc., 1989.

[67] A. R. Conn, S. Le Digabel, Use of quadratic models with mesh-adaptive direct search for constrained black box optimization, Optimization Methods and Software 28 (1) (2013) 139–158.

[68] S. M. Wild, R. G. Regis, C. A. Shoemaker, Orbit: Optimization by radial basis function interpolation in trust-regions, SIAM Journal on Scientific Computing 30 (6) (2008) 3197–3219.

[69] M. Schonlau, W. J. Welch, D. R. Jones, Global versus local search in constrained optimization of computer models, Lecture Notes-Monograph Series (1998) 11–25.

[70] S. Wright, J. Nocedal, Numerical optimization, Springer Science 35 (1999) 67–68.

[71] P. E. Gill, E. Wong, Methods for convex and general quadratic programming, Mathematical Programming Computation 7 (1) (2015) 71–112.

[72] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, Journal of Global optimization 13 (4) (1998) 455–492.

[73] S. R. Alimo, P. Beyhaghi, T. B. Bewley, Implementation of dense lattices to accelerate delaunay-based optimization for linear constraint domains., Optimization Methods and Software Journal, Under preparation.

[74] S. Alimohammadi, P. Beyhaghi, T. Bewley, Delaunay-based optimization in cfd leveraging multivariate adaptive polyharmonic splines (maps), in: 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2017, p. 0129.

[75] R. B. Gramacy, H. K. H. Lee, Bayesian treed gaussian process models with an application to computer modeling, Journal of the American Statistical Association 103 (483) (2008) 1119–1130.

[76] J. Xie, P. I. Frazier, S. Sankaran, A. Marsden, S. Elmohamed, Optimization of computationally expensive simulations with gaussian processes and parameter

uncertainty: Application to cardiovascular surgery, in: Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on, IEEE, 2012, pp. 406–413.

[77] J. Snoek, H. Larochelle, R. P. Adams, Practical bayesian optimization of machine learning algorithms, in: Advances in neural information processing systems, 2012, pp. 2951–2959.

[78] J. H. Conway, N. J. A. Sloane, Sphere packings, lattices and groups, 1998.

[79] V. J. Torczon, Multi-directional search: a direct search algorithm for parallel machines, Ph.D. thesis, Citeseer (1989).

[80] V. Torczon, On the convergence of pattern search algorithms, SIAM Journal on optimization 7 (1) (1997) 1–25.

[81] A. Zhigljavsky, A. Zilinskas, Stochastic global optimization, Vol. 9, Springer Science & Business Media, 2007.

[82] B. O. Shubert, A sequential method seeking the global maximum of a function, SIAM Journal on Numerical Analysis 9 (3) (1972) 379–388.

[83] D. R. Jones, C. D. Perttunen, B. E. Stuckman, Lipschitzian optimization without the lipschitz constant, Journal of Optimization Theory and Applications 79 (1) (1993) 157–181.

[84] A. Alexandrov, Convex polyhedra, gosudarstv, Izdat. Tehn.-Teor. Lit., Moscow-Leningrad.

[85] J. Nocedal, S. Wright, Numerical optimization, Springer Science & Business Media, 2006.

[86] P. Belitz, Applications on multi-dimensional sphere packings: derivative-free optimization.

[87] P. Belitz, T. Bewley, Efficient derivative-free optimization, in: Decision and Control, 2007 46th IEEE Conference on, IEEE, 2007, pp. 5358–5363.

[88] A. L. Marsden, M. Wang, J. E. Dennis, P. Moin, Optimal aeroacoustic shape design using the surrogate management framework, Optimization and Engineering 5 (2) (2004) 235–262.

[89] H.-M. Gutmann, A radial basis function method for global optimization, Journal of Global Optimization 19 (3) (2001) 201–227.

[90] B. Pooriya, B. Thomas, A derivative-free optimization algorithm for the efficient minimization of time-averaged statistics., Journal of Computational Optimization and Applications. Under review.

[91] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans, Reconstruction and representation of 3d objects with radial basis functions, in: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, 2001, pp. 67–76.

[92] P. E. Gill, W. Murray, M. A. Saunders, Snopt: An sqp algorithm for large-scale constrained optimization, SIAM review 47 (1) (2005) 99–131.

[93] M. Fuhry, L. Reichel, A new tikhonov regularization method, Numerical Algorithms 59 (3) (2012) 433–445.

[94] J. D. Riley, Solving systems of linear equations with a positive definite, symmetric, but possibly ill-conditioned matrix, Mathematical Tables and Other Aids to Computation (1955) 96–101.

[95] M. Drela, Xfoil: An analysis and design system for low reynolds number airfoils, in: Low Reynolds number aerodynamics, Springer, 1989, pp. 1–12.

[96] A. L. Marsden, Optimization in cardiovascular modeling, Annual Review of Fluid Mechanics 46 (2014) 519–546.

[97] S. Sankaran, C. Audet, A. L. Marsden, A method for stochastic constrained optimization using derivative-free surrogate pattern search and collocation, Journal of Computational Physics 229 (12) (2010) 4664–4682.

[98] P. Luchini, M. Quadrio, A low-cost parallel implementation of direct numerical simulation of wall turbulence, Journal of Computational Physics 211 (2) (2006) 551–571.

[99] A. S. Beyhaghi, Pooriya, T. Bewley, A multiscale, asymptotically unbiased approach to uncertainty quantification in the numerical approximation of infinite time-averaged statistics., Under preparation.

[100] T. A. Oliver, N. Malaya, R. Ulerich, R. D. Moser, Estimating uncertainties in statistics computed from direct numerical simulation, Physics of Fluids (1994-present) 26 (3) (2014) 035101.

[101] S. T. Salesky, M. Chamecki, N. L. Dias, Estimating the random error in eddy-covariance based fluxes and other turbulence statistics: the filtering method, Boundary-layer meteorology 144 (1) (2012) 113–135.

[102] C. Foias, M. Jolly, O. Manley, Kraichnan turbulence via finite time averages, Communications in mathematical physics 255 (2) (2005) 329–361.

[103] M. Jalali, V. K. Chalamalla, S. Sarkar., On the accuracy of overturn-based estimates of turbulent dissipation at rough topography, J. Phys. Oceanogr. submitted.

[104] D. Goluskin, Bounding averages rigorously using semidefinite programming: mean moments of the lorenz system, arXiv preprint arXiv:1610.05335.

[105] B. Talgorn, S. Le Digabel, M. Kokkolaras, Statistical surrogate formulations for simulation-based design optimization, Journal of Mechanical Design 137 (2) (2015) 021405.

[106] Y. Bazilevs, L. Beirao da Veiga, J. A. Cottrell, T. J. Hughes, G. Sangalli, Isogeometric analysis: approximation, stability and error estimates for h-refined meshes, Mathematical Models and Methods in Applied Sciences 16 (07) (2006) 1031–1090.