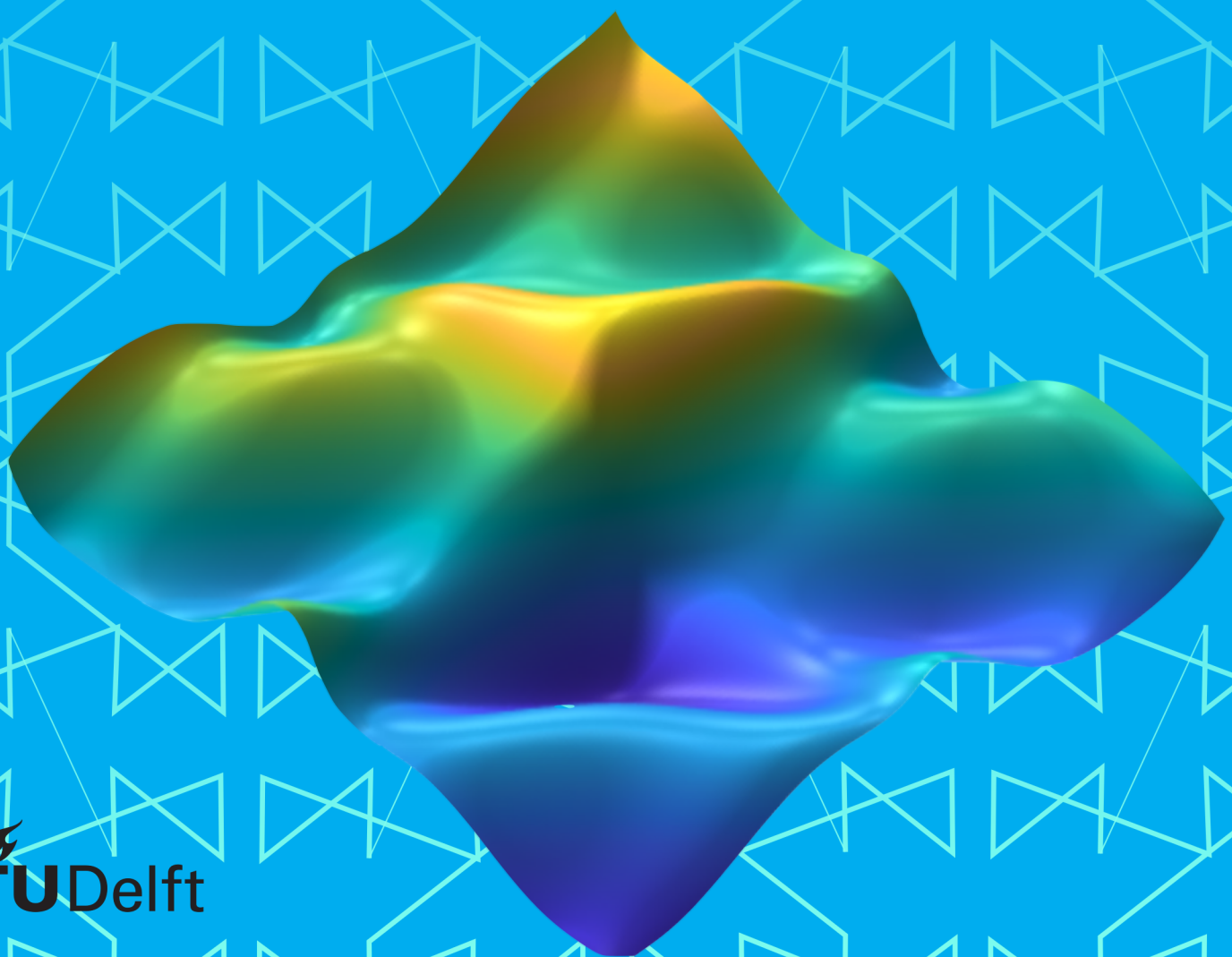
Distributed and Asynchronous Algorithm for Smooth High-dimensional Function Approximation using Orthotope B-splines

J. Meyer

Delft University of Technology



TU Delft

Distributed and Asynchronous Algorithm for Smooth High-dimensional Function Approximation using Orthotope B-splines

by

J. Meyer

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Wednesday February 24th, 2021 at 13:30.

Student number:	4303113	
Project duration:	January 18th, 2020 – February 24th, 2021	
Thesis committee:	Dr.ir. C.C. de Visser,	TU Delft, supervisor
	Dr. Z. Al-Ars,	TU Delft,
	Prof.dr.ir. M. Mulder,	TU Delft, chair

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Summary

Loss-of-control in-flight (LOC-I) is the leading cause of fatalities in aviation and is a category that encompasses all accidents where the flight crew is unable to maintain control of the aircraft in-flight.

System identification plays two key roles in mitigating accidents related to LOC-I. The first is to generate accurate models of aircraft during upsets and stalls for training. Performing the training in simulators enables the simulation to be performed at a lower cost and significantly safer. The second role that system identification plays is in adaptive flight control. Adaptive flight control enables the control systems to adapt to changes in the aircraft aerodynamic model. This allows the pilot to more easily handle the aircraft despite these changes.

Smooth aerodynamic models are important for some of the most widely used adaptive flight control algorithms, such as nonlinear dynamic inversion and incremental nonlinear dynamic inversion. Additionally, chattering in the control inputs can reduce the lifetime of components due to fatigue and wear. This last point is especially of interest for mass-produced aircraft with lower quality components, such as those found in most commercial quadrotors, which has seen a recent rise in public interest. Moreover, these quadrotors are also operating in public spaces, where safety is of paramount importance.

This thesis presents an approach for smooth high-dimensional function approximation that is robust to ill-conditioning and can handle low data coverage that is typical in high-dimensional spaces. Oblique projections and first-order optimisation with early stopping are critical for this robustness. The presented algorithm can be optimised asynchronously and within each asynchronous operation, all operations can be executed in parallel. Furthermore, a look has been taken at the memory system and the organisation of the data in memory to improve cache performance. Finally, a conflict-free organisation of shared memory is presented to enable the algorithm to be effectively executed in parallel.

Preface

This thesis develops a novel Distributed and Asynchronous B-spline (DAB) algorithm for smooth high-dimensional function approximation. It also looks at various computational aspects, such as memory organisation for efficient and fast execution of the algorithm. The thesis is divided into two main sections: the scientific paper and the literature study. The thesis is concluded with the conclusions of the research and recommendations for further research. The research is aimed at efficient approximate methods to applying boundary constraints for splines but may have further use for other fields with sparse constraints.

This master thesis is the final milestone of my double degree in Aerospace and Computer Engineering.

I would like to express my gratitude to my family and Gabriella Wiersma for supporting me during this endeavour. Finally, I would like to thank Coen de Visser for his guidance and discussions through this final phase of the master degree.

J. Meyer
Delft, February 2021

Contents

1 Introduction **1**

1.1 Research Objectives & Questions 2

1.2 Outline 3

I Research Paper **5**

2 Introduction **8**

3 Literature **10**

3.1 Function Approximation 10

3.2 Optimisation 12

3.3 Optimisation in the Context of Multivariate B-splines 13

4 Background **14**

4.1 Mutlivariate Orthotope B-splines 14

4.2 Kaczmarz's Method 19

4.3 Caches 19

5 DAB Algorithm **21**

6 Oblique Projections **24**

7 Orthogonal Constraints **27**

8 Memory Layout **29**

8.1 Morton Z-ordering 29

8.2 Bank Conflict Avoidance 31

9 Conclusions & Recommendations **33**

10 References **34**

11 B-spline Directional Derivatives **36**

12 B-coefficient Deduplication **36**

13 Performance Analysis **37**

II Literature Survey **41**

14 Function Approximation **43**

14.1 Regression vs Interpolation . . . 43

14.2 Overview of Function Approximation Methods . . . 44

14.2.1 Multivariate Simplex B-splines. . . . 45

14.2.2 Triangulation . . . 47

14.2.3 Constraints . . . 47

14.2.4 Directional Derivatives . . . 49

14.3 Multivariate Simplotope B-Splines. . . . 49

14.3.1 Constraints . . . 51

15 Optimisation **53**

15.1 Convex Programming. . . . 53

15.2 Cost Functions . . . 54

15.3 Separability . . . 55

15.4 Derivative-free Methods . . . 55

15.5 Gradient-based Methods . . . 56

15.5.1 Matrix conditioning . . . 56

15.5.2 Bisection Search . . . 57

15.5.3 Steepest Descent . . . 57

15.5.4 Conjugate Gradient Descent . . . 57

15.5.5 Sequential Linear Programming . . . 58

15.5.6 Newton's Method . . . 58

15.5.7 Quasi-Newton Methods. . . . 59

15.6 Constrained Optimisation Approaches . . . 60

15.6.1 Kaczmarz Method . . . 60

15.6.2 Null-space Projection . . . 60

15.6.3 Lagrange Multiplier Methods . . . 61

15.6.4 Mesh Smoothing . . . 64

15.7 Online Optimisation Algorithms 64
15.7.1 Warm-starting 65
15.7.2 Online Gradient Descent 65
15.7.3 Recursive Least Squares 65
15.8 Multi-grid Methods 66

16 Conclusions **67**

17 Recommendations **69**

Bibliography **71**

Nomenclature

Function Approximation

$\hat{d}$	The number of parameters in the simplotope or simplex.
$\mathbb{M}_d^{n+1}$	The set of multi-indices in the space $\mathbb{N}_0^{n+1}$ subject to the constraint that the multi-index norm is equal to d.
$\mathbf{P}^{d,d-m}(b)$	The De Casteljau Matrix.
$\lvert \kappa \rvert$	The norm of the multi-index: $\kappa_0 + \kappa_1 + \cdots + \kappa_n$
$\mathcal{T}$	A tessellation is a set of connected splines
$\vec{1}$	Column vector of ones.
b^κ	$b_0^{\kappa_0} b_1^{\kappa_1} \cdots b_n^{\kappa_n}$
B_κ^d	The basis function of degree d associated with the multi-index κ (see Eq. 14.5)
d	The degree of the simplotope or simplex.
$D_u^m p(b)$	The directional derivative of the polynomial p in the direction of the vector u.
H	The constraint matrix.
m, $\mathbb{C}$	The order of desired continuity.
n	The number of dimensions in Cartesian space.
r	The number of continuity constraints.
t_j	A simplex t_j is an element of the tessellation $\mathcal{T}$
B-net	The spatial structure of the B-coefficients
KKT	This refers to Karush-Kuhn-Tucker optimality conditions and the matrix generated from them.
LLS	Linear Least Squares
OLS	Ordinary Least Squares

Optimisation

α	Step-size parameter in gradient descent/ascent approaches.
$\mathcal{L}$	Lagrangian of the cost function
λ	Dual variable in the Lagrangian method
$\mathbb{R}^n$	An n-dimensional set of real numbers.
ρ	Penalty parameter for penalty-based methods.
θ	Parameters of the function approximator to be optimised.
$\mathbb{N}_0^n$	An n-dimensional set of whole numbers.
c	Equality constraint vector
H	Equality constraint matrix
MSE	Mean Squared Error

1

Introduction

Loss-of-control in-flight (LOC-I) is the leading cause of fatalities in aviation and is a category that encompasses all accidents where the flight crew is unable to maintain control of the aircraft in-flight. This includes stalls/upsets, structural failure and adverse meteorological conditions [33]. In the period from 2009 to 2018, IATA determined that there were in total 64 LOC-I incidents in general aviation, 94% of which involved fatalities [33]. Moreover, it resulted in the highest number of fatalities compared to any other accident category, accounting for around 60% of all fatalities in general aviation [33]. In a similar study by Boeing for the same time period for commercial air transport, LOC-I was found to be the cause of around 48% of all onboard fatalities [6]. This has led to the Federal Aviation Authority (FAA) and the European Aviation Safety Agency (EASA) to mandate that flight crew are trained in handling and preventing stalls and upsets since 2019 [23] and 2016 [21], respectively. These changes have also been reflected in the best practices and guidance material by the International Air Transport Association (IATA) [32]. Furthermore, in the explanatory note by EASA, they cite LOC-I as one of the agency's highest priorities [22].

System identification plays two key roles in mitigating accidents related to LOC-I. The first is to generate accurate models of aircraft during upsets and stalls for training. Performing the training in simulators enables the simulation to be performed at a lower cost and significantly safer. The second role that system identification plays is in adaptive flight control. Adaptive flight control enables the control systems to adapt to changes in the aircraft aerodynamic model. This allows the pilot to more easily handle the aircraft despite these changes. Typically, control algorithms require knowledge of the model of the aircraft to determine the gains to be used to guarantee a certain level of performance. Nonlinear dynamic inversion (NDI) is the most used approach that is useful for nonlinear control of an aircraft. An inaccurate nonlinear model means that the nonlinear dynamics will not be sufficiently neutralised and the linear controller will not perform as desired [14]. Therefore having an accurate model at all times is critical, especially after significant changes to the model. Global smooth models are particularly important for NDI and Incremental NDI (INDI) as it ensures that there are no discontinuities in the control inputs [14] and control effectiveness Jacobian. The discontinuities can to an extent be damped by the aircraft's inertia and actuator dynamics; however, in highly manoeuvrable aircraft, the damping may not be sufficient and may induce aeroelastic flutter. Alternatively, chattering can reduce the lifetime of components due to fatigue and wear. This last point is especially of interest for mass-produced aircraft with lower quality components, such as those found in most commercial quadrotors.

With the advent of air taxi projects like Uber Air[1] and CityAirbus[2], the need for adaptive control will be even more evident as these aircraft will be flying over densely-populated regions and the estimated loss of life will increase significantly. Furthermore, projects like ATTOL[3] aim for fully autonomous

[1] https://www.uber.com/us/en/elevate/uberair/
[2] https://www.airbus.com/innovation/zero-emission/urban-air-mobility/cityairbus.html
[3] https://www.airbus.com/innovation/future-technology/autonomy.html#projects

operations which will require accurate models as little to no human intervention will be possible.

One specific challenge for system identification is over-actuated systems with significant aerodynamic interactions. The reason this is challenging is that increasing the number of independent variables of the regression problem increases the problem size exponentially - this is typically referred to as the curse of dimensionality or, in this case, combinatorial explosion. This also increases the amount of data required to fit a model without overfitting the data. Traditional adaptive control is implemented via gain-scheduling, which is affected by this problem to a greater extent and is not a fault-tolerant solution. In order to make the problem more manageable with hardware restraints, two approaches are common: approximating a high-dimensional space with an abstract low-dimensional space, and parallelism. These abstract spaces are in most cases arbitrary and are not a unique nonlinear mapping, limiting the interpretability of the controller. Neural networks are an example of black-boxes and are agnostic to the underlying representation, and are therefore viable options; however, nonlinear representations are often unstable [46] and, typically, there are no guarantees on performance when learning and adapting. Consequently, certification, verification, and validation are significantly harder in comparison to traditional control approaches [34].

1.1. Research Objectives & Questions

The goal of this thesis is the development of a robust system identification method with applications to high-dimensional and online model identification.

> "The aim of this research is to contribute to safer future forms of air transport, like delivery drones and air taxis, with the aid of robust adaptive flight control systems by investigating constrained optimisation algorithms for smooth high-dimensional and online system identification that can be executed in a parallel and asynchronous manner."

To investigate the feasibility and the applicability of the selected constrained optimiser and to achieve the main goal of this research, several research questions will be stated.

> What is the impact of the hyper-parameters on the optimiser's accuracy on the validation dataset and how does this compare to Newton's method?

The first research question is void of computational aspects and is mainly aimed at determining how well the optimiser can fit to the training data and how this compares to the optimal constrained analytical solution given by Newton's method. Additionally, it is also necessary to assess the optimiser accuracy when data coverage is low - which is common in high-dimensional space - as it leads to overfitting of the model to the limited data and results in worse validation accuracy.

> Does the quality of the unconstrained input to the constrained optimiser affect the accuracy of the algorithm?

This builds on the issue of overfitting as most constrained optimisers are composed of steps with unconstrained optimisers and the quality of the input can therefore constrain the maximum achievable accuracy of the constrained optimiser.

> Can oblique projections improve model accuracy on the validation dataset when compared to orthogonal projections when data coverage is low?

The next research question is aimed at improving the accuracy and robustness of the algorithm by ensuring that constraints are weighted by the level of certainty or amount of data in a region of the

domain. This avoids well-estimated regions of the domain being negatively impacted by the smoothness constraints. This aim of robustness also targets the aforementioned problem of low data coverage.

> Can duplicated B-coefficients along boundaries be eliminated when applying constraints?

When formulating the regression problem with splines, model parameters at the edges where two splines meet are duplicated and makes the model more local. This is useful for solving the unconstrained problem and for evaluating the model locally; however, when applying the constraints they result in some redundant computation that - if eliminated - can improve the speed of the algorithm. The duplicated model parameters also increase the memory footprint of the model.

> Can the theoretical cache performance be improved by resorting the coefficients in the tessellation?

Modern hardware is built using a memory hierarchy to reduce cost and improve performance; however, ensuring that the necessary data can be found in the fastest level is based on the cache policy of the processor and the cache transaction size. Transactions between levels of hierarchy transfer batches of data that are spatially local in memory. If the access patterns of the optimiser are regular and predictable, it is possible to sort the data such that data that is frequently accessed together is also stored together.

> Can orthogonal constraints be used to improve the speed of the algorithm at the cost of more memory usage?

Orthogonal constraints increase the speed of convergence of optimisers as the search directions are orthogonal and do not impact the results from previous search directions; however, they typically come at the cost of reduced sparsity and can increase therefore significantly increase the required amount of computation.

> Can bank conflicts in shared memory on the GPU be avoided to improve performance of the algorithm?

The final research question is aimed at determining whether it is possible to use parallel hardware effectively to execute the algorithm. Since the constraints are between model parameters in different directions, it is necessary to ensure that the parameters can be read from memory in parallel for each direction. If this is not possible, the algorithm cannot be effectively executed in parallel and will approximate a sequential algorithm as memory accesses will become sequential.

1.2. Outline

This thesis will be structured into two main parts: the research paper and the literature survey.

The research paper's layout will now be explained. The first part of the research paper involves an introduction to the field and the problem at hand. Thereafter, a terse summary of the literature survey is given and how the literature fits in the context of B-splines. Next, the required background information is given on multivariate B-splines, including how to ensure smoothness between the splines, and Kaczmarz's method is explained from two complementary perspectives. Subsequently, an overview and justification of the components of the Distributed Asynchronous B-spline (DAB) algorithm is given. Each component of the DAB algorithm is then decomposed and explained in further depth with results to support the components purported purpose. Finally, the research paper closes with the conclusions and recommendations for further research.

The literature survey is divided into two main sections: function approximation and optimisation. The function approximator dictates which optimisers are suitable and the structure of the problem - such as sparsity- and is thus treated first. First, it is explained why regression was chosen opposed to interpolation. Thereafter, the pros and cons of several function approximators is given. Finally, some background on the selected function approximator is given. Next, an overview of the optimisation problem is given and the structure of the optimisation problem. The chapter is then divided into unconstrained, constrained, and online optimisation.

The thesis then concludes with the conclusions and recommendations for further research. This chapter builds upon the conclusions and recommendations of the research paper and links them back to the research questions in the prior section.

I

Research Paper

Distributed and Asynchronous Algorithm for Smooth High-dimensional Function Approximation using Orthotope B-splines

J. Meyer · C.C. de Visser

Abstract Aircraft are complex systems with, in some cases, high-dimensional non-linear interactions between control surfaces. When a failure occurs, adaptive flight control methods can be utilised to stabilise and make the aircraft controllable. Adaptive flight control methods, however, require accurate aerodynamic models - where first-order continuity is necessary for estimating the control derivatives and mitigating chattering that can reduce the longevity of components. Additionally, high-dimensional offline model identification with current approaches can take several hours for a few dimensions and this means model iterating and hyper-parameter tuning is often not feasible. Current approaches to smooth high-dimensional functional approximation are not scalable, require global communication between iteration steps, and are ill-conditioned in higher dimensions. This research develops the Distributed Asynchronous B-spline (DAB) algorithm that is more robust to ill-conditioning, due to low data coverage, by using first-order methods with acceleration and weighted constraint application. This algorithm is also suitable for continuous state-spaces. Smooth aerodynamic models can be determined in exactly $n \cdot \hat{r}$ iterations, where $\hat{r}$ is the number of continuity equations in a single dimension and n is the number of dimensions. Moreover, memory reorganisation is proposed to avoid false sharing and conflict-free use of shared memory on the GPU to ensure that the algorithm runs efficiently in parallel.

Keywords Smooth High-dimensional Linear Function Approximation · Linearly-constrained Convex Quadratic Programming · Asynchronous Algorithms · Parallel Algorithms · Multivariate B-spline · System Identification · GPU · Cache-aware

---

J. Meyer
Control & Simulation Divsion, Faculty of Aerospace Engineering, Kluyverweg 1, 2629HS Delft, The Netherlands
E-mail: johann.meyer@gmail.com

C.C. de Visser
Control & Simulation Divsion, Faculty of Aerospace Engineering, Kluyverweg 1, 2629HS Delft, The Netherlands

1 Introduction

Loss-of-control in-flight (LOC-I) is the leading cause of fatalities in aviation and is a category that encompasses all accidents where the flight crew is unable to maintain control of the aircraft in-flight. This includes stalls/upsets, structural failure and adverse meteorological conditions [21]. In the period from 2009 to 2018, IATA determined that there were in total 64 LOC-I incidents in general aviation, 94% of which involved fatalities [21]. Moreover, it resulted in the highest number of fatalities compared to any other accident category, accounting for around 60% of all fatalities in general aviation [21]. In a similar study by Boeing for the same time period for commercial air transport, LOC-I was found to be the cause of around 48% of all onboard fatalities [5]. This has led to the Federal Aviation Authority (FAA) and the European Aviation Safety Agency (EASA) to mandate that flight crew are trained in handling and preventing stalls and upsets since 2019 [15] and 2016 [13], respectively. These changes have also been reflected in the best practices and guidance material by the International Air Transport Association (IATA) [20]. Furthermore, in the explanatory note by EASA, they cite LOC-I as one of the agency's highest priorities [14].

System identification plays two key roles in mitigating accidents related to LOC-I. The first is to generate accurate models of aircraft during upsets and stalls for training. Performing the training in simulators enables the simulation to be performed at a lower cost and significantly safer. The second role that system identification plays is in adaptive flight control. Adaptive flight control enables the control systems to adapt to changes in the aircraft aerodynamic model. This allows the pilot to more easily handle the aircraft despite these changes. Typically, control algorithms require knowledge of the model of the aircraft to determine the gains to be used to guarantee a certain level of performance. Nonlinear dynamic inversion (NDI) is the most used approach that is useful for nonlinear control of an aircraft. An inaccurate nonlinear model means that the nonlinear dynamics will not be sufficiently neutralised and the linear controller will not perform as desired [35]. Therefore having an accurate model at all times is critical, especially after significant changes to the model. Global smooth models are particularly important for NDI and Incremental NDI (INDI) as it ensures that there are no discontinuities in the control inputs [35] and the control effectiveness Jacobian. The discontinuities can to an extent be damped by the aircraft's inertia and actuator dynamics; however, in highly manoeuvrable aircraft, the damping may not be sufficient and may induce aeroelastic flutter. Alternatively, chattering can reduce the lifetime of components due to fatigue and wear. This last point is especially of interest for mass-produced aircraft with lower quality components, such as those found in most commercial quadrotors.

With the advent of air taxi projects like Uber Air[1] and ATTOL by Airbus[2], the need for adaptive control will be even more evident as these aircraft will be flying over densely-populated regions and the estimated loss of life will increase significantly. Furthermore, projects like ATTOL aim for fully autonomous operations which will require accurate models as little to no human intervention will be possible.

[1] https://www.uber.com/us/en/elevate/uberair/

[2] https://www.airbus.com/innovation/future-technology/autonomy.html#projects

One specific challenge for system identification is over-actuated systems with significant aerodynamic interactions. The reason this is challenging is that increasing the number of independent variables of the regression problem increases the problem size exponentially - this is typically referred to as the curse of dimensionality or, in this case, combinatorial explosion. This also increases the amount of data required to fit the model without overfitting to the data. Traditional adaptive control is implemented via gain-scheduling, which is affected by this problem to a greater extent and is not a fault-tolerant solution. In order to make the problem more manageable with hardware restraints, two approaches are common: approximating a high-dimensional space with an abstract low-dimensional space, and parallelism. These abstract spaces are in most cases arbitrary and are not a unique nonlinear mapping, limiting the interpretability of the controller. Neural networks are an example of black-boxes and are agnostic to the underlying representation, and are therefore viable options; however, nonlinear representations are often unstable [29] and, typically, there are no guarantees on performance when learning and adapting. Consequently, certification, verification, and validation are significantly harder in comparison to traditional control approaches [22].

High-performance computing and general-purpose GPU (GPGPU) computing have enabled problems to continue scaling despite power issues due to increasing clock frequencies, which was one of the primary approaches to increasing single-core performance in the beginning of computing. Brain simulation and genome sequencing have seen dramatic performance improvements due to GPGPU computing and the parallel computing paradigm, see for example the implementations in [33] for brain simulation and [28] for sequencing. Other fields related to aerospace, such as fluid dynamics and structural analysis, have also seen large improvements with GPGPU computing.

The scope of this research will be limited to function approximation of a continuous state-space where the locations of measurements are also a function of time. These choices change the structure of the optimisation problem and prevent some matrices, such as the regressor matrix, from being factorised offline. The constraints are tied to the tessellation, which is not adaptive in this case and is therefore fixed.

The aim of the research is to develop a fast algorithm for smooth online system identification that is robust to ill-conditioning by using oblique projections to simplify and improve constraint application with limited data - which is of particular concern in high-dimensional spaces. It is paramount to improve the computation speed and complexity of the constrained optimisation step to ensure the feasibility and scalability of the algorithm in online and adaptive flight control settings with limited computational power and therefore these aspects have been taken into account in this research.

The paper will first look at the current state-of-the-art in function approximators and optimisation. Thereafter some background information on multivariate B-splines, the continuity equations between the splines, and the Kaczmarz algorithm will be given. Following, a high-level overview of the Distributed Asynchronous B-spline (DAB) algorithm will be explained. Finally, the key elements of the algorithm will be expanded in the subsequent section with their complementary results.

2 Literature

This section is divided into two main parts: the first focusses on function approximators and the second on optimisation. The second part is further subdivided into two sections: unconstrained and constrained optimisation.

2.1 Function Approximators

System identification is a special subset of function approximation with the goal of modelling dynamical systems. Dynamical systems can be modelled based on first principles or they can be modelled using black-boxes, which generate an arbitrary input-output mapping that is not intended to be interpreted. Interpretability, analysability, and robustness is of key importance for certification of control systems [22]. Black-boxes also do not enable the use of a priori information when developing the model which can be used to improve accuracy and/or simplify the model complexity. The most common function approximators are fuzzy-blending, tensor-product splines, thin-plate splines, artificial neural networks, and multivariate B-splines.

First, it is necessary to distinguish between interpolation and regression. Interpolation means the function approximator must pass through all data points where as regression finds the best fit without this requirement. An implicit assumption in interpolation is that the measured data is noise-free, which is an assumption that is often wrong in experimental data and is particularly erroneous when estimating the derivative of the output. Moreover, it requires sufficient data to uniquely generate the model whereas regression can use regularisation alleviate this problem. Furthermore, in higher dimensions the amount of data required for a unique solution becomes infeasible, especially one that avoids overfitting.

Fuzzy-blending is a manual approach to smoothing and connecting splines that requires expert knowledge [35]. The smoothing must also be reperformed if the granularity of the grid is changed. Moreover, higher-order continuity cannot be achieved, which is of importance for computing the control effectiveness Jacobian. Additionally, manual approaches quickly become infeasible when the dimensionality of the problem increases.

Tensor-product splines requires taking the tensor-product of 1D polynomials. The downside of these tensor-products is that the tensor-product can create precision errors when working in higher dimensions. But most importantly, the data needs to be collected on a rectangular grid [35] and for some applications, such as flight data which is scattered, this is not possible nor desirable.

Thin-plate splines is another approach that can, in contrast to tensor-product splines, handle scattered data. The downside to thin-plate splines is that it requires a global radial basis function for each data point [35]. As a result of the global nature of the basis functions, all basis functions are required to evaluate and optimise the spline, which can quickly become computationally infeasible. This is exacerbated as the dataset increases in size and dimensions.

Artificial Neural Networks (ANN) are a set of algorithms that are modelled loosely on the neural system. These algorithms are among the most popular in recent times

due to the power and potential these algorithms have shown in domains such as reinforcement learning [30] and computer vision [27], especially when it comes to generalisation. The benefit of generalisation (extrapolation) may be of limited value for highly nonlinear aircraft and should be investigated separately. The major problems with ANNs are that they are non-convex, black-box systems, computationally dense, and require modifications to ensure that continuity and smoothness is satisfied. By computationally dense, it is meant that even if the outputs of the network are sparse, the entire network needs to be evaluated to determine the output. Spiking Neural Networks are a new generation of neural networks for neuromorphic hardware that will be able to exploit this sparsity. Moreover, since they are black-box systems, the entire ANN needs to evaluated multiple times to estimate derivative information which can be computationally expensive depending on the size of the network. Furthermore, small changes in the input do not necessarily mean small changes in the output as a result of the nonlinearities. It is also much harder to ensure all constraints are satisfied throughout the entire domain as they suffer from problems such as catastrophic forgetting (see [29] for more details). Catastrophic forgetting is also more apparent for online learning as the data is highly correlated. For example in reinforcement learning, nonlinear function approximators, like neural networks, are unstable as a result of this and have required modifications like memory buffers of past data and copies of the networks to stabilise the algorithms [30]. The non-convexity of ANNs lead to slower training and gradient-based methods cannot guarantee global optimality; however, first-order algorithms and local minima are often sufficient in practice. The size of the networks and the plethora of hyper-parameters that are associated with them are empirically determined and provide no guarantees of their performance in-situ and complicates the design and certification process [22].

Multivariate Simplex and Simplotope B-Splines are linear-in-the-parameters function approximators that can be used to model nonlinear data. They can be solved using Linear Least Squares and can therefore be optimised using convex solvers. Since it is a linear-in-the-parameters function approximator, there are more guarantees for the stability of the approximator and the mathematical tools for analysis of the approximator are more mature. The curse-of-dimensionality is present since no nonlinear map is used. It is, nevertheless, possible to use dimensionality reduction techniques, such as autoencoders [17], principle component analysis, tensor networks [24], and hashing functions to alleviate this but, as with any compression technique, some data loss will occur, accuracy will be decreased, and sparsity will be reduced. The extent to which this occurs can be managed with appropriate trade-offs. An alternative approach to reducing the problem size, is using null-space projection [9] of the constraint matrix; however, this approach still reduces the sparsity. Nonetheless, the interpretability is lost, if a dimensionality reduction technique is used. Moreover, it should remain static to avoid invalidating the function approximator that relies on that input representation. The curse-of-dimensionality is additionally abated by the use of local basis functions, as these result in a very sparse problem structure that can be efficiently evaluated. The optimisation of the parameters can also occur in a distributed and parallel fashion [1]. Continuity and differential constraints can be applied as shown by De Visser [35] and a priori knowledge can be incorporated using physical constraints as shown by van der Peijl [34] and Huisman [19].

2.2 Optimisation

Optimisation is a large field and a wide body of literature exists. The optimisation problem for function approximation is typically limited to the Least Absolute Error (LAE), Least Squared Error (LSE), and Huber Loss. LSE is the most popular for regression and quadratically penalises the error. LSE has a smooth cost function and has an analytical solution. LAE is useful for sparse solutions but requires the use of linear iterative solvers that can handle non-smooth cost functions [3]. An additional benefit of LAE is that outliers are not as heavily penalised as for LSE and is therefore more robust. Huber loss is a hybrid of LSE and LAE and aims to improve the robustness of LSE and the convergence rate of LAE; however, it requires switching between two different optimisers.

2.2.1 Unconstrained Optimisation

Unconstrained optimisation can be split into two main methods: gradient-based and derivative-free methods. Derivative-free methods are useful when the loss function is noisy or when evaluating the gradient is expensive or impossible. Additionally, they are useful for non-convex optimisation as they can escape local minima. Gradient-based methods are typically preferable as the gradient information can be used to direct the optimisation and lead to faster convergence, especially in higher dimensions where the curse of dimensionality can make it infeasible to effectively explore the domain. Moreover, the gradient information can be used to determine the optimality of the solution. Gradient-based methods convergence rates are sensitive to the scaling of the loss function and as a result matrix conditioning is important. gradient descent and coordinate descent are the two foundational algorithms when looking at gradient-based methods. Most algorithms are variations of these two where the changes are the step size and the direction of descent. Popular gradient-based algorithms are steepest descent, Newton's method, quasi-Newton methods, conjugate gradient descent, and momentum-based gradient descent. Momentum-based methods - such as ADAM [26], Nesterov accelerated gradient [31], RMSprop, and AdaGrad [10] - are popular in machine learning and increase the rate of convergence significantly but may reduce generalisation accuracy [25]. Newton's method results in one-step convergence but requires the inversion of a matrix, which can be slow and numerically inaccurate as well as being badly conditioned in higher dimensions.

2.2.2 Constrained Optimisation

Constrained optimisation are often meta-algorithms in the sense that they require unconstrained optimisation algorithms to be executed during some of the steps, which direct the unconstrained optimisation to a constrained optimal solution.

Kaczmarz method is a special case of Projection onto Convex Sets (POCS), where the method of alternating projections is applied to affine constraints. In the field of image reconstruction, it is referred to as the algebraic reconstruction technique. Dykstra's method is a generalisation of Kaczmarz for convex sets [11] and translated

convex sets [12] and guarantees that the algorithm converges to the orthogonal projection within the constraint set. A parallel version of Dykstra's algorithm is given by Gaffke and Mathar [16]. Kaczmarz method can also be used as an unconstrained optimisation method for solving the least squares problem; however, it excels when the problem is sparse as in the constraints between splines. It should be noted that when used for constrained optimisation, the algorithm does not take into account the loss function and does not result in the constrained optimal solution. It does; however, introduce some flexibility into the algorithm by enabling the algorithm to be split into two orthogonal steps: cost optimisation and constraint satisfaction.

Null-space projection uses a projection to transform a constrained problem into an unconstrained problem. The null-space projection can be computed offline; however, it is susceptible to numerical precision errors. Additionally, the Hessian of the unconstrained problem is no longer block-diagonal, which significantly increases the cost of solving the unconstrained problem. A numerically robust online form of null-space projection is presented by Zhu [39], where the constraints are applied by differing initial conditions.

Lagrange multiplier methods are a family of algorithms that introduces the concept of dual variables and the dual function. The duality gap is the difference between the primal and dual function and can be used to draw conclusions about the optimality of the constrained solution [6]. The most intuitive version is the penalty method that involves adding a penalty term for violated constraints to the loss function. The penalty multiplier is iteratively increased until it converges to the constrained optimal solution. This approach can sometimes lead the cost function to focus on constraint satisfaction over optimising the original cost function. For quadratic problems the Lagrange multiplier method has an analytical solution; however, computationally this method is poorly conditioned and slow due to the large matrix inversion. An implicit iterative method by [2] is often faster and does not explicitly store the Lagrange multipliers. Dual ascent, dual decomposition, and the method of multipliers alternate between optimising the primal and dual functions. Dual decomposition is the parallelised version of dual ascent. Dual ascent is more sparse than the method of multipliers but requires a smooth and differentiable cost function. The method of multipliers introduces an additional hyper-parameter - the penalty factor - and it enables non-differentiable cost functions with possibly infinite values to be used [6]. The alternating direction method of multipliers(ADMM) makes the problem separable by adding auxiliary constraints and an additional optimisation step [6]. Additional optimisation steps can only be added if the steps are orthogonal [7]; however, this also means these steps can be run in parallel. Wei proposed a distributed [37] and an asynchronous [38] version of ADMM; however, these have yet to be applied to multivariate B-splines.

2.3 Optimisation in the Context of Multivariate B-splines

The first use of multivariate B-splines for aerodynamic modelling was by De Visser [35] and he used the recursive least squares formulation by Zhu [39]. Dual ascent has been used for multivariate simplex B-splines by De Visser in a sequential [9] and distributed manner [8]; however, that was for wavefront reconstruction, which

involves continuous measurements at fixed locations and enables a different set of assumptions to be made on the structure of the problem. A distributed version of ADMM has been used for multivariate simplotope B-splines by van den Aarssen [1] and Hooij [18] for aerodynamic model identification. Hooij expanded on van den Aarssen's work by introducing a distance weighting function for using pre-failure data in online model identification. De Visser [8] and van den Aarssen [1] both utilise a hybrid approach where the grid is split into partitions and null-space projection is used internally in a partition and another approach is used for inter-partition constraints. The disadvantage of null-space projections is that they result in matrix fill-in and are numerically unstable for high dimensions and large partition sizes. Awanou [2] used the implicit Lagrangian iterative solver for solving partial differential equations using multivariate B-splines. Karagöz's [24] tensor-network B-splines are equivalent to simplotope B-splines by Van Den Aarssen [1] and multiplex B-splines by Visser [36]. Karagöz's [24] approach to high-dimensional function approximation is a form of compression of the network to reduce its memory footprint at the cost of accuracy; however, at the same time the approximator is no longer interpretable and continuity is not taken into account. This research aims to improve the robustness of constrained online optimisation while ensuring that the algorithm can be effectively executed in an online setting.

3 Background

The background is divided into three main sections: theory on multivariate orthotope B-splines, Kaczmarz's algorithm, and caches.

3.1 Multivariate Orthotope B-splines

A simplex t_j is a geometrical structure that minimally spans a set of n dimensions and is composed of exactly $n + 1$ vertices where the edges are given by simplices of one dimension lower. Simplices can be connected to each other at their edges. A collection of these connected simplices is called a tessellation $\mathcal{T}$.

Each simplex has a local coordinate system known as the barycentric coordinate system. A useful property of barycentric coordinates is that points that lie outside the simplex have one or more negative barycentric coordinates. To convert Cartesian coordinates to barycentric coordinates, the transformation matrix $T \in \mathbb{R}^{n\times n}$ for a simplex should be formed. The barycentric coordinates $b \in \mathbb{R}^{n+1}$ can then be computed using Eq. 2 and Eq. 3. The vertices v of the simplex are numbered using the superscript and their Cartesian coordinates are indexed by their subscript.

$$T = \begin{bmatrix} v_0^1 - v_0^0 & \cdots & v_0^n - v_0^0 \\ \vdots & \vdots & \vdots \\ v_{n-1}^1 - v_{n-1}^0 & \cdots & v_{n-1}^n - v_{n-1}^0 \end{bmatrix} \tag{1}$$

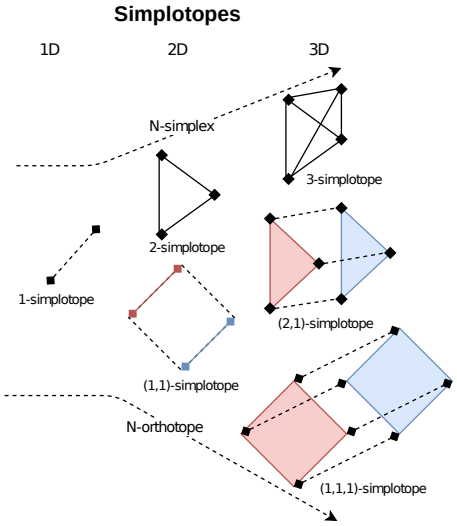


Fig. 1 Simplotopes can be described as N-simplices and N-orthotopes at the extremes. The figure is a 2D projection of the N-dimensional simplotopes. N-simplices are generated by adding a vertex in the n^{th} dimension. N-orthotopes are created by duplicating the (N-1)-orthotope in the n^{th} dimension and connecting the respective vertices or, alternatively, as a tensor-product of 1D-simplices. Adapted from [1].

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = T^{-1} \begin{bmatrix} x_0 - v_0^0 \\ x_1 - v_1^0 \\ \vdots \\ x_{n-1} - v_{n-1}^0 \end{bmatrix} \tag{2}$$

$$b_0 = 1 - b_1 - b_2 - \cdots - b_n \tag{3}$$

The Bernstein basis functions are given by the multinomial theorem (Theorem 1). It is a stable basis that sums to 1. Additionally, it is local because the basis functions are zero outside of the simplex. Each basis function B_κ^d is scaled by its B-coefficient c_κ to approximate any polynomial, as given by De Boor's theorem(Theorem 2). $\mathbf{B}^d \in \mathbb{R}^{l \times \hat{d}}$ is the matrix formulation of the summation in Eq. 7, where l is given by the number of barycentric coordinates being used and $\hat{d}$ is given by Eq. 4. In the case of the theorem, $l = 1$. $\vec{1}$ is a column vector of ones. The B-coefficients each belong to a specific spline t_j and may be distinguished by a superscript to indicate this membership. The barycentric coordinate of each B-coefficient with respect to a spline of degree d is given by Eq. 10. $\kappa \in \mathbb{N}_0^{n+1}$ is a multi-index. An additional notation used for multi-indices is $\mathbb{M}_d^{n+1}$, where the superscript denotes the set of $\mathbb{N}_0^{n+1}$ subject to the constraint that the norm of the multi-index is equal to the subscript d and the number of elements in the set is determined by filling in the subscript in Eq. 4.

$$\hat{d} = \frac{(d+n)!}{d!n!} \tag{4}$$

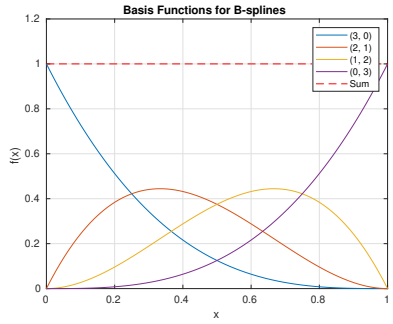


Fig. 2 Bernstein basis functions for a 3rd-order 1D spline.

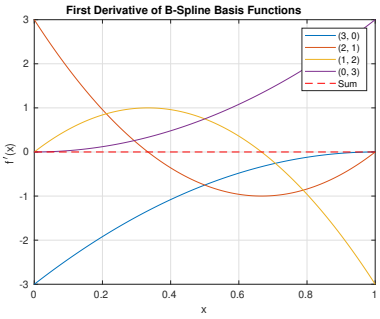


Fig. 3 1st-order derivative of the Bernstein basis functions for a 3rd-order 1D spline.

Theorem 1 (Multinomial Theorem for Barycentric Coordinates)

$$(b_0 + b_1 + \cdots + b_n)^d = \sum_{|\kappa|=d} \frac{d!}{\kappa_0!\kappa_1!\cdots\kappa_n!} b_0^{\kappa_0} b_1^{\kappa_1} \cdots b_n^{\kappa_n} \tag{5}$$

$$= \sum_{|\kappa|=d} \frac{d!}{\kappa!} b^{\kappa} \tag{6}$$

$$= \sum_{|\kappa|=d} B_{\kappa}^{d}(b) \tag{7}$$

$$= \mathbf{B}^d \vec{1} \tag{8}$$

$$= 1 \tag{9}$$

Theorem 2 (De Boor's Theorem) *Any polynomial $p(x)$ of degree d can be written as a sum of basis functions B_{κ}^{d} scaled by its B-coefficient $c_{\kappa}^{t_j}$:*

$$p(x) = \sum_{|\kappa|=d} c_{\kappa}^{t_j} B_{\kappa}^{d}(b(x))$$

$$b(c_{\kappa}) = \frac{\kappa}{d} \tag{10}$$

Since the basis functions are local to a spline, the evaluation of the multivariate B-spline for any point only requires one to evaluate the function approximator for the spline that contains the point. During regression, updates only need to be applied locally. For second order methods, this manifests itself as a sparse block-diagonal structure which significantly reduces the computational complexity, especially for large tessellations.

Simplotope B-splines - also known as multiplex B-splines in [36] - is a modification to simplex B-splines where the tensor-product of lower-dimensional orthogonal simplices are used instead of one high-dimensional simplex. The more layers in a simplotope spline, the fewer the number of constraints for the same grid size; however, the number of parameters per simplotope also increases with the number of layers. More parameters per simplotope means larger block sizes in the block diagonal data matrix.

The benefit of more layers is that the total number of coefficients are reduced, as fewer coefficients are duplicated and there are fewer simplotopes in the same triangulation and therefore fewer constraints to be applied.

Simplotope equations similar to their simplex counterparts and are denoted with π to distinguish them. The multi-index κ is similar to the those for simplices; however, instead of one multi-index, there is one for each simplex layer l, which are concatenated to form κ (i.e. $\kappa = [\kappa_1, \cdots, \kappa_l]$). The same applies to the barycentric coordinates b. Eq. 11 is used to evaluate a simplotope. B_{κ_i} in Eq. 12 is the B-form for the simplex in layer i with the subset of the multi-index κ for that layer.

$$\pi(b) = \sum_{\substack{|\kappa_i|=d_i \\ \forall i \in [1,l]}} c_\kappa B_\kappa(b) \tag{11}$$

$$B_\kappa(b) = \prod_{i=1}^{l} B_{\kappa_i}^{d_i}(b_i) \tag{12}$$

Orthotope B-splines is a special case of simplotope B-splines, where the lower-dimensional simplices are all one dimensional (see Fig. 1). Orthotopes are also referred to as hyperrectangles in literature. Orthotope B-splines are beneficial as they simplify computation of neighbouring splines, high-dimensional tessellations, spline membership, and indexing. Additionally, orthotopes have fewer neighbours than their equivalent simplex form and, in higher dimensions, simplices become *sliver simplices* - as described by De Visser [35] - which are difficult to fill with data. Moreover, the approximation power of 1-dimensional simplices is higher than that of higher-dimensional simplices as couplings are created between the dimensions when applying continuity. The use of 1-dimensional splines also enables the tensor-product to be evaluated in parallel.

See De Visser [35] for more detailed information on multivariate simplex B-splines and Van den Aarssen [1] for more details on simplotope B-splines.

3.1.1 Constraints

There are two main types of constraints that are applicable to aerodynamic function approximation. The first being continuity constraints between the simplices in the tessellation and the second being a priori knowledge of the problem.

Continuity constraints are particularly important for control systems that are based on the estimated aerodynamic model as discontinuities or also lack of differentiability between simplices will either limit the control algorithms that can be applied or limit the performance of the control system. The discontinuities and lack of differentiability is also an artifact of using splines and not necessarily related to the underlying function that is being approximated. $v_*^{t_i}$ is the barycentric coordinate of the out-of-edge vertex for simplex t_i with respect to simplex t_j. The non-zero multi-index for the out-of-edge vertex for simplex t_i is denoted as $\kappa_*^{t_i}$. Finally, γ is a multi-index. A visual depiction of the coefficients involved in the constraints for the 1D case is given in Fig. 19. It should be noted that the continuity matrix is purely a function of the tessellation and its connectivity and is not dependent on the data being regressed. This means that this

matrix is always constant. A consequence of applying the spline continuity constraints is that the local model becomes a global model. The distance the constraints propagate through the model is related to the number of degrees of freedom of each simplex [35]. Therefore, higher-order models with lower-order continuity leads to more local models. The upper-bound on the number of constraints is given by Eq. 13 multiplied by the number of shared edges. This is also the number of constraints generated by following the algorithm in Theorem 3 and, as a result, some constraints are redundant which has repercussions for some solvers. If the nullity of the constraint matrix is zero, only the trivial solution exists.

Theorem 3 (Continuity Equations)

$$
\begin{aligned}
& c^{t_i}_{\kappa_0,\cdots,\kappa_n} = \sum_{|\gamma|=m} c^{t_j}_{(\kappa_0,\cdots,\kappa_n)+\gamma} B^m_\gamma(v^{t_i}_*), \qquad \{\forall m \in \mathbb{N}_0 | 0 \leq m \leq r\} \\
\textit{subject to:} \quad & \kappa^{t_i}_0 + \cdots + \kappa^{t_i}_n = d \\
& \kappa^{t_i}_* = m \\
& \kappa^{t_j}_* = 0
\end{aligned}
$$

$$
\hat{r} = \sum_{m=0}^{r} \frac{(d-m+n-1)!}{(d-m)!(n-1)!} \tag{13}
$$

The application of constraints for simplotopes is very similar to that of simplex B-splines; however, requires a slight modification. The process of generating the continuity constraint as in Theorem 3 is only applied to the simplex that contains the out-of-edge vertex. The tensor-product is then applied to all permutations of the simplices in the other layers. The number of continuity constraints for an edge of a simplotope is given by Eq. 14, where $\hat{r}_*$ is given by Eq. 13. $\hat{r}_*$ is the number of constraints for the out-of-edge simplex. $\hat{d}_i$ and $\hat{d}_*$ are the number of parameters in layer i and the out-of-edge simplex, respectively. The propagation of constraints in orthotope B-splines can be more easily controlled than in simplex B-splines due to the more regular structure.

$$
\hat{r}_\pi = \hat{r}_* \frac{\prod_{i=1}^{l} \hat{d}_i}{\hat{d}_*} \tag{14}
$$

A priori knowledge can for example be incorporated into the function approximation problem by including constraints due to physical constraints associated with the aircraft. However, the simplex B-splines operate using barycentric coordinates and, in order to be able to apply these constraints, the physical constraints in Cartesian coordinates need to be transformed to barycentric constraints. This approach to constraint application is referred to as physical splines in literature [19] and [34]. It is also possible to apply constraints based on the directional derivatives as in [9] using the De Casteljau matrix in Appendix A.

3.2 Kaczmarz's Method

Kaczmarz's method [23] optimises the cost functions given by Eq. 15 and Eq. 16 when A is overdetermined and underdetermined, respectively. θ_0 are the initial model parameters and y is the vector of target values. The iterative equation for orthogonal projections is given by Eq. 17. A_j is the j$^{\text{th}}$ row of A. j can be selected in any order but is commonly chosen either stochastically or cyclically. Note γ is an optional relaxation parameter.

The method can be viewed from two perspectives. The first perspective is that it involves iteratively projecting a vector onto a linear constraint set - i.e. it is a special case of projection onto convex sets (POCS). The second perspective is that of gradient descent, where the numerator is the gradient of the cost function and the optimal step-size for the i$^{\text{th}}$ iteration α_i^* is given by Eq. 18. The algorithm, in general, does not converge in n steps, where n is the number of rows in the matrix A. This is because the rows are not guaranteed to be orthogonal nor are the search directions A-orthogonal, as is the case in the conjugate gradient method[32]. The algorithm is computationally beneficial when the rows of A are sparse making it cheaper and more precise than a line search. Additionally, the solution using Kaczmarz's method for an underdetermined system results in a higher precision than other iterative approaches. This is of importance, for example, when applying equality constraints.

$$\underset{\theta}{\text{minimise}} \; ||y - A\theta||_2^2 \tag{15}$$

$$\begin{aligned} &\underset{\theta}{\text{minimise}} \; ||\theta - \theta_0||_2^2 \\ &\text{subject to } A\theta = y \end{aligned} \tag{16}$$

$$\theta_i = \theta_{i-1} + \gamma \frac{(y_i - A_j\theta_{i-1})A_j^T}{||A_j||_2^2} \tag{17}$$

$$\alpha_i^* = \frac{1}{||A_j||_2^2} \tag{18}$$

3.3 Caches

The fact that processor performance improves much faster with time than memory performance is known as the processor-memory performance gap. Memory hierarchies is one approach to mitigate the loss of performance due to this gap and works by organising system memory into a hierarchy of faster but progressively smaller and more expensive memories. Data that is accessed more frequently or more recently is placed in the faster levels to reduce latency. The data from slower main memory is transferred to the faster caches in blocks of contiguous data known as cache lines to reduce routing complexity. If the data is not reused, these blocks will be evicted or, if the data is not contiguous, cache pollution may occur - reducing the effective size of the cache. Cache thrashing occurs when the cache is too small to hold the required data leading to data being evicted before it can be reused.

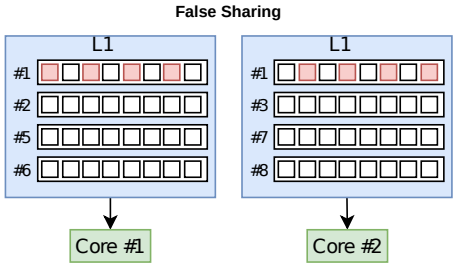


Fig. 4 A simplified cache system is illustrated, where the # denotes a location in main memory. The red squares indicate the memory addresses that will be modified by the core. Whenever an element in a cache line is written to, the other caches are informed by the write-invalidation policy that it must re-fetch its cache line from main memory or the lowest shared cache level. False sharing is the case where only the green elements are written to by a core. Even though no relevant data has changed for the core, the data must be re-fetched leading to a large latency.

In multicore systems with multiple caches, cache coherency needs to be ensured. Cache coherency is the process of ensuring all caches agree on the same value for a memory address. First, it is necessary to ensure that all data in a cache is eventually written to main memory. The two main approaches to ensuring this is the write-through and write-back protocols. Write-through is simpler and makes all writes to a memory address in the cache immediately to the copy in main memory. This approach is typically found in GPUs and has the disadvantage of using a lot of possibly unnecessary memory bandwidth, if many writes occur. Write-back is more complex and typically found in CPUs. It involves keeping track of whether the data has been modified and defers writing to main memory until the cache block is evicted or requested by another core. If another core requires the data, this approach leads to a longer latency as data must first be written to main memory before it can be requested by the other cache. One common approach to cache coherency is using write-invalidation. Write-invalidation involves broadcasting to all cache copies that the data that they have for a cache line is no longer valid and should be reloaded from main memory. This leads to another problem, which is known as false sharing. False sharing is when different cores are using different data in the same cache lines (see Fig. 4). When either core writes to the data, the other core needs to reload the data from main memory, even though its data has not been modified.

Scratchpad memory is a form of fast memory that is used to store intermediate calculations that do not need to be stored in main memory. It is typically - but not necessarily - explicitly managed by a programmer. It is often similar to L1 cache; however, it does not require the overhead of cache coherency protocols or writing to main memory as the data is local to a core. This reduces contention for the memory bandwidth. In Nvidia GPUs, the scratchpad memory is known as shared memory and is sometimes shared with the L1 cache and for the Turing and Ampere architectures it can be set to maximum size of 64KB and 100KB, respectively. The scratchpad memory is local to a core for CPUs or Streaming Multiprocessor (SM) for GPUs (see Fig. 5) and cannot be used for inter-core communication.

[4] Intra-warp communication is available using warp-level primitives.

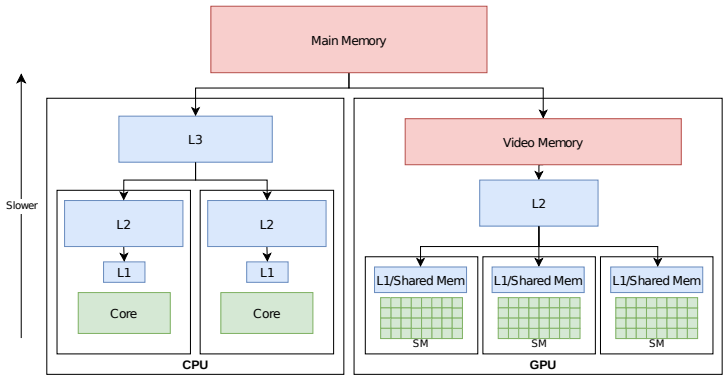


Fig. 5 Each rectangle indicates a physical separation between resources. In red is RAM, in blue is cache and green is used for computing cores. CPUs are based on a smaller number of large complex cores. In contrast, GPUs are based on a large number of simpler cores that are grouped into Streaming Multiprocessors (SM) - in Nvidia terminology - that share certain resources, like an L1 cache. Each of the tiny cores in an SM can only communicate with other cores within its SM using shared memory or with all cores via main memory.[4]

Sorting the data in memory according to expected access patterns leads to the expected latency being closer to the cache latency rather than the main memory latency, which can differ by several orders of magnitude. Additionally, avoiding false sharing is critical to avoid unnecessary cache invalidation. Finally, effective use of scratchpad memory can reduce the load on the memory bandwidth allowing relevant data to be retrieved with a lower latency.

4 DAB Algorithm

In this section an overview of the key features of the Distributed Asynchronous B-spline Algorithm (DAB) will be given. A graphical overview of the algorithm is given in Fig. 6. First, a maximum allowable continuity is set to limit continuity propagation. Thereafter the reasoning behind oblique projections is explained. Finally, the need for a specific memory layout is given.

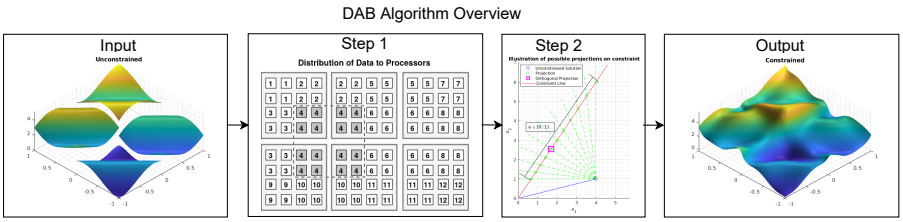


Fig. 6 An overview of the DAB algorithm. The algorithm takes as input an unconstrained solution and distributes the data of each vertex to a different processor. The constraints are then applied to each vertex through oblique projections. The output of the algorithm is a constrained solution.

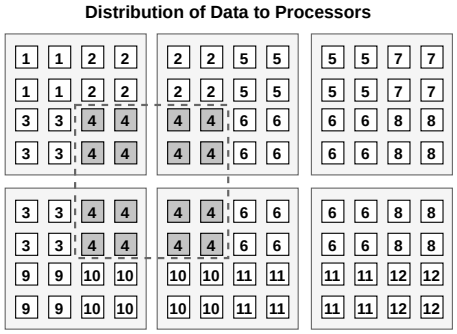


Fig. 7 Each vertex (indicated by dashed line) can be distributed to a different processor (indicated by number). The example is of a third-order B-spline in two dimensions. In higher dimensions, the blocks become orthotopes.

The maximum continuity of a spline of degree d is given by Eq. 19. Graphically, this means the B-coefficients involved in the continuity cannot pass the halfway mark with respect to the spline for that layer or the continuity will propagate throughout the entire tessellation. This propagation has two negative impacts on the algorithm. The first is that the continuity will take time to propagate from one side of the tessellation to the other. This can be abated by multi-grid methods (see for example[4]); however, it is computationally better to keep the modifications in the model local. Additionally, this is physically more intuitive and justifiable. Moreover, the parallelisability of the algorithm is impacted as the algorithm must either utilise red-black ordering or include auxiliary constraints to ensure correctness. While not necessary, the algorithm simplifies significantly if the degree of the model is set based on the continuity required using Eq. 19 as edge cases are eliminated (see the boundaries of Fig. 7). The edge cases can also be alleviated by padding the domain with splines with their weights set to 0. This may, however, lead to slower convergence than handling the edge cases directly. The algorithm is asynchronous in that each vertex requires no communication with any other vertex, as shown in Fig. 7. This is beneficial in the offline/batch setting and when large high-dimensional tessellations are being used. Within each vertex, the algorithm is parallel but requires all the projections for a direction to complete before changing direction.

$$\mathbb{C}^{\max} = \left\lfloor \frac{d+1}{2} \right\rfloor - 1 \tag{19}$$

The B-spline cost function is given by Eq. 20. Gradient descent with ADAM [26] with early stopping is used to optimise the unconstrained problem for fast convergence with low data coverage, which can impact the stability of second-order algorithms or otherwise lead to overfitting. Kaczmarz's method is used to apply the constraints and is solving Eq. 16, with θ_0 given by the unconstrained optimal solution. Kaczmarz's method gives a solution that satisfies all constraints but does not provide the optimal constrained solution as it does not take into account the regressor matrix. Kaczmarz's method is equivalent to dual ascent with the second term of the derivative being changed from the Hessian $B^T B$ to the identity matrix (compare Eq. 21 and Eq. 22).

This means the cost achieved by Kaczmarz's method is not cost-optimal based on the available data B. It is, however, possible to reduce the cost by using a few steps of dual ascent prior to applying Kaczmarz's method. With more dual ascent iterations, the solution will approach the optimal constrained solution. Lagrange multipliers can be interpreted as "forces" applied to the B-coefficients. When the Lagrange multipliers are large, it is indicative of overfitting the model to the obtained data or that the desired continuity level is not appropriate for the current model. If the data is not the issue, then the latter issue can be improved by reducing the size of the orthotopes or by reducing the desired order of continuity. Moreover, dual ascent is a global approach as the basis functions (see Fig. 2) are non-zero away from vertices and need to take into account the data from the entire orthotope with either 2 or more dimensions, or first-order continuity. The constrained solution obtained by Kaczmarz's method is the one with the minimum "forces" applied and results in less oscillatory behaviour at apparent discontinuities between splines. The oscillatory behaviour is akin to the Gibb's phenomenon. The discontinuities are termed as apparent as overfitting to limited data can result in pseudo discontinuities, which may not be representative of the true underlying model. These discontinuities are indistinguishable from actual discontinuities until sufficient data is available and therefore modelling them may or may not be desirable.

$$\begin{aligned} &\underset{\theta}{\text{minimise}} \ ||y - B\theta||_2^2 \\ &\text{subject to } H\theta = 0 \end{aligned} \tag{20}$$

$$\frac{\partial f}{\partial \theta} = (B^T B)^{-1} B^T y - I H^T \lambda \tag{21}$$

$$\frac{\partial f}{\partial \theta} = (B^T B)^{-1} B^T y - (B^T B)^{-1} H^T \lambda \tag{22}$$

Oblique projections are used to extrapolate or weight the estimates of the B-coefficients such that poorly-estimated coefficients do not influence the well-estimated coefficients when applying continuity. Fig. 8 illustrates this by showing how the right spline pulls the left splines coefficients down. Fig. 9 illustrates how oblique projections avoids this and achieves better accuracy on the training dataset and, in principle, the validation dataset.

The optimisation algorithm applies the constraints by iterating through the out-of-edge dimensions sequentially. Memory on parallel computing systems divide the memory into multiple memory chips to enable parallel memory access; however, when the out-of-edge dimension changes, the naive ordering (Fig. 10) results in all parallel threads reading from the same memory chip and requires each access to be sequentially handled. By padding the memory, each thread can be guaranteed parallel access to its data.

Additionally, memory on computing systems are organised in a hierarchical fashion from fastest and most expensive to slowest and cheapest - where the difference can be several orders of magnitude. To make efficient use of the hierarchy, data is often transferred in contiguous batches to the faster memory. If all memory accesses are closer together, better use of the memory hierarchy is made. Sorting the B-coefficients

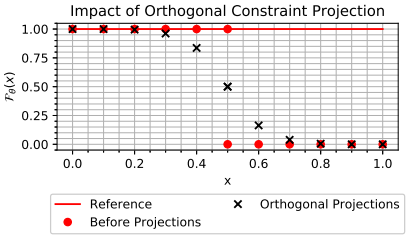


Fig. 8 The function has been regressed for $x < 0.5$ on the reference function. For orthogonal projections, it can be seen that the predictions from the left orthotope are pulled down to match the right orthotope, even though no data has been acquired for $x > 0.5$.

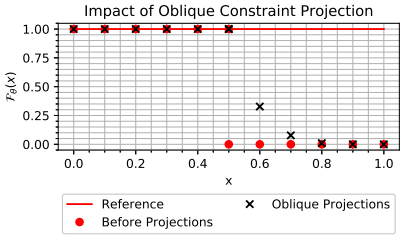


Fig. 9 The function has been regressed for $x < 0.5$ on the reference function. For oblique projections, it can be seen that the predictions from the right orthotope are pulled up to match the left orthotope, thereby improving the accuracy on the reference function.

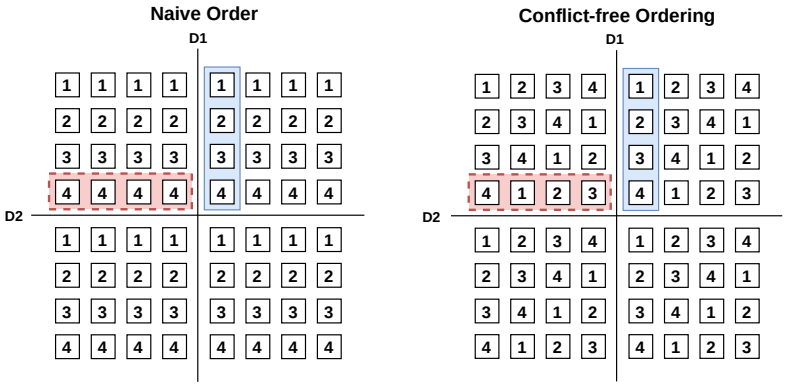


Fig. 10 Two methods to map the B-coefficients of an orthotope to memory banks. Consecutive memory addresses are mapped to consecutive memory banks. The naive ordering results in a 4-way bank-conflict when the vertical direction is the out-of-edge direction. The conflict-free ordering pads the data and therefore uses more memory but avoids the bank conflict across the D2 line (in red) leading to parallel accesses to the B-coefficients and faster overall execution.

in memory according to ones that are frequently used together for continuity and regression can ensure that all data in the caches are relevant. In high-level programming languages, less control is available; however, rows or columns of matrices are still typically stored contiguously to minimise overhead and the cache repercussions of sharing cache lines is still present.

5 Oblique Projections

This section builds upon the view of Kaczmarz's method as an iterative projection algorithm, as mentioned in Section 3.2. The original algorithm typically assumes orthogonal projections; however, this approach is not necessarily optimal as the estimate of the parameters in a neighbouring spline may be better or worse depending on the

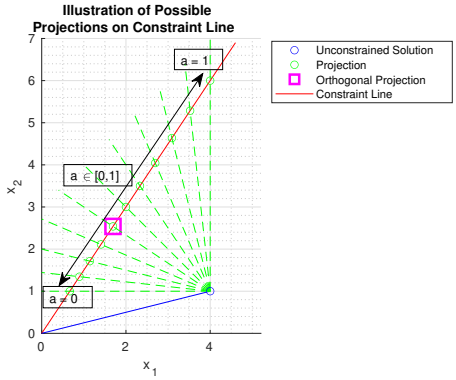


Fig. 11 The projection direction can be modified by varying the parameter a in Eq. 23, which changes the gradient v. Horizontal or vertical projection prevents a change in x_2 and x_1, respectively. This is useful to improve the robustness of the algorithm in the case that the uncertainty of one parameter is larger than another. A projection direction with $a = 0.5$ is not necessarily equivalent to the orthogonal projection direction.

data coverage. The vector v in Eq. 23 can be used to control the projection direction. A constraint is always between two splines and the relation in Eq. 23 always holds; however, in order to generalise the relation, the vectors need to be extended to the size of constraint equation and the elements are one in the left vector if the coefficients belong to the left spline and similarly for the right vector. a is then the normalised weight for the spline. The m^{th} orthogonal projection direction H_m is given by the continuity relation for m^{th}-order continuity. The rows of H are identical for the same continuity order but differ only in the indexing function. As a result, it was decided to subscript H with m instead of i (see the indexing function in Appendix C for more details). For clarity, the splines are referred to as the left and right spline with respect to the edge; however, this is a general relation for all dimensions. The projection for the m^{th} continuity order is denoted as $\mathbb{P}^m$ and is computed using Eq. 25 and Eq. 24. $\theta_k^{i_l^m, i_r^{0:m}}$ denotes the value of the model parameters at iteration k at indices i_l^m and i_r^0 through to i_r^m, in that order. The indices i use a subscript to differentiate between the parameters of the two splines and the superscript determines the offset from the edge. The operator $\odot$ indicates element-wise multiplication. The parameters of the model are updated using Eq. 26.

$$v = a \begin{bmatrix} 1 \\ 0 \end{bmatrix} - (1 - a) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{23}$$

$$w^m = \left[w_l \overbrace{w_r \cdots w_r}^{\text{m+1}} \right]^T \tag{24}$$

$$\mathbb{P}^m = \frac{H_m \theta_k^{i_l^m, i_r^{0:m}}}{H_m H_m^T} H_m^T \odot w^m \tag{25}$$

$$\theta_{k+1}^{i_l^m, i_r^{0:m}} = \theta_k^{i_l^m, i_r^{0:m}} - \frac{\mathbb{P}^m}{w_l + w_r} \tag{26}$$

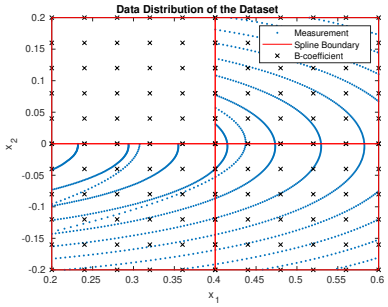


Fig. 12 This figure illustrates poor data coverage, as some splines have no data.

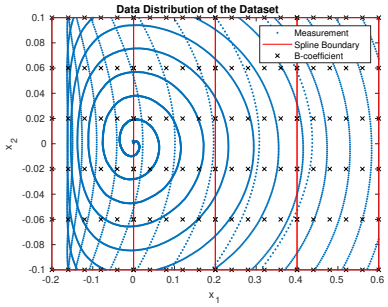


Fig. 13 This figure illustrates better data coverage, as all splines have some data.

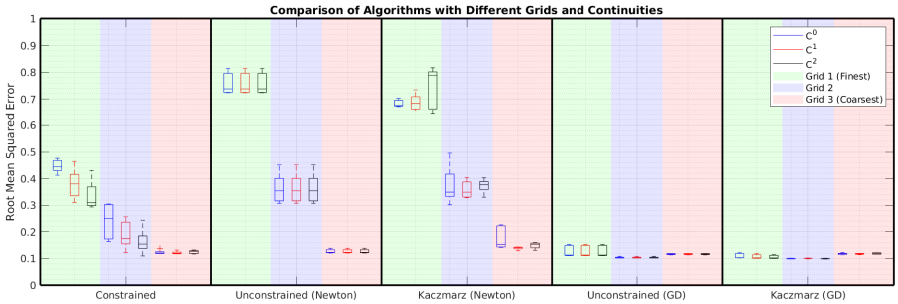


Fig. 14 This figure is generated - using 5-fold cross validation on the data presented in Fig. 13 - to illustrate how overfitting of the unconstrained solution can lead to worse validation accuracy, which cannot be solved by higher-order continuity. Continuity order has little impact on validation accuracy for the proposed algorithm.

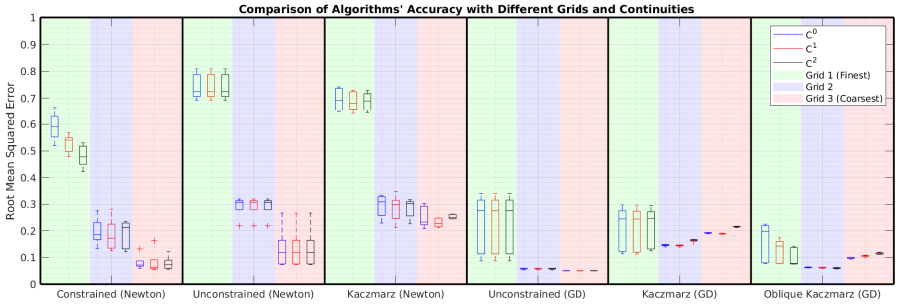


Fig. 15 This figure is generated - using 5-fold cross validation on the data presented in Fig. 12 - to illustrate how oblique projections can be used to improve validation accuracy while reducing the variance in the estimated coefficients when data coverage is low in some regions of the domain. Coarser grids are better and continuity has little impact on validation accuracy for the tested dataset. The variance in the accuracy of the model for Kaczmarz's method is generally lower and is therefore more reliable for online use.

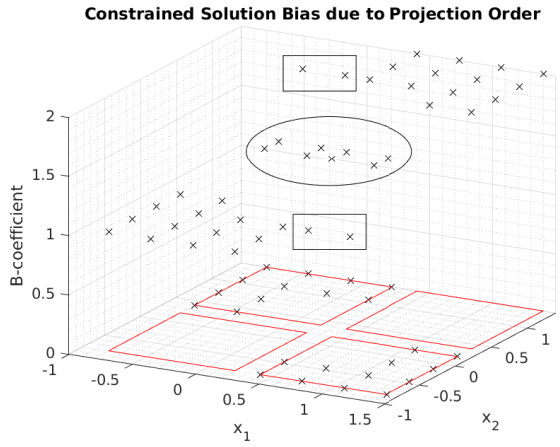


Fig. 16 This figure illustrates how the order of the projection directions can influence the constrained solution. The first projection direction is x_1. The coefficients in the rectangles illustrate that the first projection extrapolates the splines in these directions first. Projection in the x_2 direction then averages the values in the ellipse. The bias is apparent from the rectangular shape in the ellipse. If the projection order is swapped, the ellipse will be rotated by 90 degrees. It is interesting to note that oblique projections enable continuity to be propagated in the diagonal direction, which would not be possible if the constraints to neighbouring splines were to be omitted due to insufficient data.

In order for the weighting of the oblique projections to operate as intended, the projections must converge in a dimension before it can proceed to the next dimension. Orthogonal constraints, as in Section 6, provide a clear benefit as only one iteration per dimension is required. The weightings do not change and are determined by the quality of the data in the spline or by whether or not the spline contains valid or sufficient data.

6 Orthogonal Constraints

Orthogonal constraints have a geometrical interpretation in the case of spline continuity but it is first necessary to graphically review the spline continuity formulation in Theorem 3. In [35], the continuity equations are formulated asymmetrically across the spline boundary, as can be seen in Fig. 19. This asymmetry means that once $\mathbb{C}^1$ has been applied, $\mathbb{C}^0$ is no longer true. This case will, hereon, be referred to as constraint divergence. In contrast, the orthogonal formulation - as in Fig. 20 - updates both splines' parameters thereby ensuring that lower-order continuity is still maintained. For even and odd continuity-orders, the coefficients are anti-symmetric and symmetric about the spline boundary, respectively. Although not apparent from Fig. 19 and Fig. 20, the computation and number of memory accesses approximately doubles asymptotically, in the 1D case (compare Fig. 21 and Fig. 22). In the higher-dimensional case, the number of operations required for orthogonality results in a significant overhead - as the orthogonality means that all parameters for all connected splines around a vertex need to be updated to avoid constraint divergence. Using the 1D orthogonal equa-

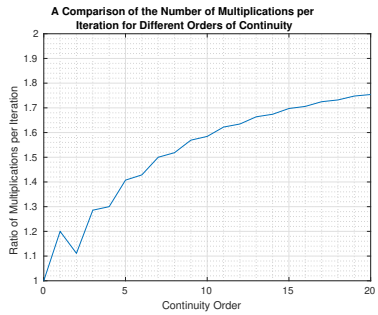


Fig. 17 The ratio of the number of multiplications required per iteration when comparing the original 1D formulation to the orthogonal 1D formulation. The ratio never exceeds two and the original version requires typically more than 1 iteration due to the non-orthogonality of the constraints. The ratio can also be deduced by comparing the number of non-zero values in Fig. 21 and Fig. 22. The jaggedness of the line arises due to the two zero values in the centre of Fig. 22 at every even continuity order. Oblique projections require convergence in the 1D case before changing direction. It is therefore never beneficial to use the original constraints in 1D.

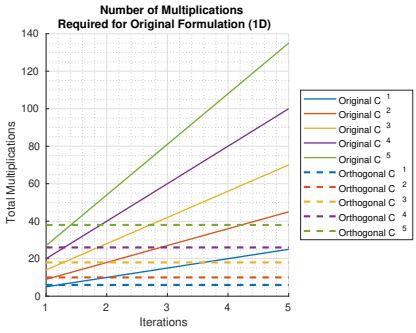


Fig. 18 The required number of iterations for convergence for the original formulation is determined by the distance of the unconstrained solution to the constrained solution, which is not known a priori. However, it is clear from the figure that the orthogonal constraints are always computationally cheaper than two iterations of the original formulation (see also Fig. 17) and the slope of the line increases with the continuity order.

tions instead of the original equations results in the algorithm converging in one pass through per dimension. The original version would need to converge in one iteration to perform better from a arithmetic complexity perspective (see Fig. 17), which is guaranteed to not occur due to non-orthogonality. Fig. 17 illustrates the ratio of the number of multiplications required per iteration for the original to the orthogonal case. The right term is for the original formulation which converges in t iterations with the per iteration complexity given by Eq. 27. The orthogonal version requires only one iteration and an upper-bound on its complexity is given by Eq. 28. The bound is an upper-bound because the 2 coefficients are zero for even continuity orders (see Fig. 22) and has been neglected from the equation. The result of Fig. 17 can also be deduced by comparing twice Eq. 27 to Eq. 28. Additionally, the memory overhead of 1D orthogonality is abated by realising that the higher-order continuity requires that the lower-order B-coefficients already be read from main memory and are therefore already stored in a register. The DAB algorithm converges in $\hat{r} \cdot n$ iterations with the algorithm's parallel arithmetic complexity given by Eq. 29. Both the original and the orthogonal formulation only need to project until convergence in each direction once (i.e. applying continuity in one direction does not impact continuity in the previous directions). It is presumed to be due to the layers being orthogonal to each other; however, further analysis is necessary.

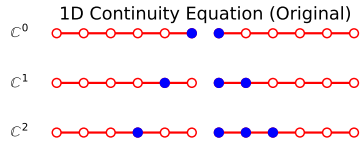


Fig. 19 The original continuity equations as stated by De Visser [35] and Theorem 3. The B-coefficients that are affected by the continuity equations are filled.

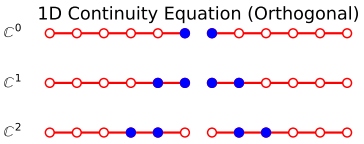


Fig. 20 The original continuity equations orthogonalised to all preceding continuity orders. The B-coefficients that are affected by the continuity equations are filled.

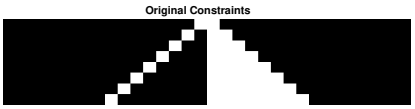


Fig. 21 The original continuity equations as stated in [35] and Theorem 3, for seventh order continuity. The B-coefficients that are affected by the continuity equations are white.

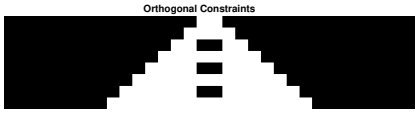


Fig. 22 The original continuity equations orthogonalised to all preceding continuity orders, for seventh order continuity. The B-coefficients that are affected by the continuity equations are white.

$$O(\text{Orig/iteration}) = \sum_{m=0}^{\hat{r}}(m+2) = \frac{\hat{r}^2+\hat{r}}{2} + 2(\hat{r}+1) \tag{27}$$

$$O(\text{Orth}) < \sum_{m=0}^{\hat{r}} 2(m+1) = \hat{r}^2 + \hat{r} + 2(\hat{r}+1) \tag{28}$$

$$O(\text{DAB}) \approx O(\hat{r}^2 n) \tag{29}$$

7 Memory Layout

This section will describe the two modifications required for better use of the memory of a system to ensure parallel execution of the algorithm.

7.1 Morton Z-ordering

Morton Z-ordering is a space-filling curve that is often used for organising data in GPUs to optimise accesses for spatial locality. The main aim of this subsection is to mitigate cache issues such as false sharing and cache pollution, as mentioned in Section 3.3. Eliminating these issues is critical for using the cache hierarchy effectively and is beneficial on both CPUs and GPUs.

With orthotope B-splines, there are 3 main z-orderings: orthotope-centred degree-based (OD) ordering, orthotope-centred continuity-based (OC) ordering, and vertex-centred continuity-based (VC) ordering. Orthotope-centred means all coefficients belonging to a orthotope are stored in a contiguous section of memory. In contrast,

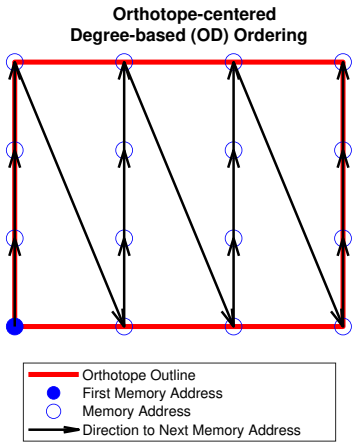


Fig. 23 The indexing starts at the blue dot and follows the arrows. This figure illustrates orthotope-centred degree-based (OD) ordering in 2D.

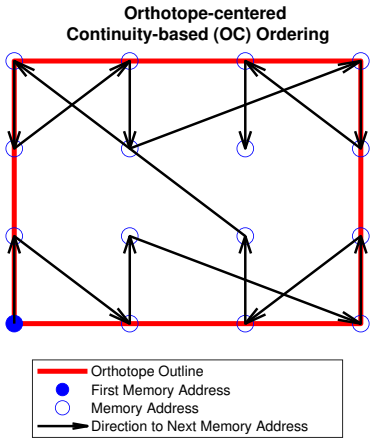


Fig. 24 The indexing starts at the blue dot and follows the arrows. This figure illustrates orthotope-centred continuity-based (OC) ordering in 2D.

vertex-centred stores all coefficients belonging to a vertex in contiguous memory. Vertex-centred ordering has the potential to save up to $2^n - 1$ global memory transactions when applying the constraints, where n is the number of dimensions. However, this benefit is mitigated by the decrease in performance when determining the unconstrained optimal solution and when evaluating the spline. Due to the resorting, extracting the data for a given spline becomes more complicated as the data is no longer contiguous. The benefit of vertex-centred ordering is further diminished when using shared memory on the GPU, as the read from global memory only needs to be performed once per kernel call. OD ordering is ordering the coefficients based on the multinomial coefficients of the spline and is identical to the ordering presented in [35] (see Fig. 23). OC ordering begins at each vertex and then moves toward the centre of the orthotope (see Fig. 24). All coefficients belonging to a vertex of an orthotope are stored contiguously improving cache performance (less cache pollution) on both the CPU and GPU when applying the constraints, enabling better memory access coalescing, and simplifying the indexing function. OC is expected to outperform OD because false sharing is avoided - as the cache lines are no longer shared between two vertices in an orthotope. False sharing is particularly detrimental to performance as the data needs to be read and written to after each projection. No quantitative analysis of the impact of false sharing on performance is available, as a full implementation of the algorithm has yet to be made. It should be noted that an additional padding may be necessary for the OC ordering to ensure two neighbouring vertices within an orthotope do not share cache lines. A non-performance benefit is that the indexing function is symmetric about its edges and means offsets are always positive away from the edge. The offset for each direction is always a power of the number of continuity equations $\hat{r}$.

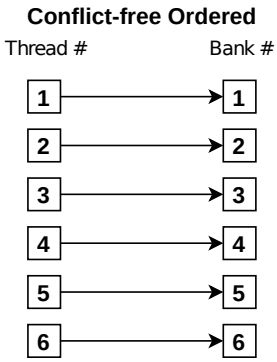


Fig. 25 Each thread accesses a sequential memory bank.

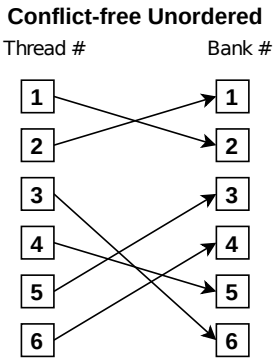


Fig. 26 Each thread accesses a different memory bank. There is no performance benefit compared to Fig. 25.

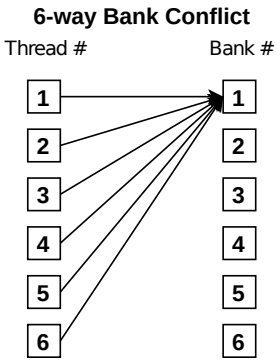


Fig. 27 All threads access the same memory bank. If the accesses are to different addresses, the accesses are performed sequentially.

7.2 Bank Conflict Avoidance

An additional memory consideration for GPUs, Field-programmable Gate Arrays (FPGAs), Application Specific Integrated Circuits (ASICs), and parallel hardware in general is that the memory can be split into multiple separate chips and is often referred to as memory banks. Each memory bank can be accessed in parallel enabling a higher data throughput. Additionally, it ensures that parallel threads do not need to stall for sequential accesses to memory. Typically, the data in the memory banks is striped, similar to RAID 0, meaning sequential memory addresses are mapped to different memory banks. Modern Nvidia GPUs, work with 32 threads in parallel and have 32 memory banks and, ideally, that means each thread has its own memory bank; however, as the data is striped, it is possible that two or more threads need to access the same memory bank and this is referred to as a bank conflict (see Fig. 27). Fine-grained control on avoiding bank conflicts is only provided by using shared memory on Nvidia GPUs and has the additional benefit of reducing memory bandwidth contention[5] when performing writes after each projection step of the algorithm. The total amount of shared memory required to store all the data for a vertex for different numbers of dimensions and continuities is given in Table 1. A first-order model in 7 dimensions is the limit for shared memory in modern hardware. B-coefficient deduplication, as in Appendix B, has the potential to increase this limit, if a conflict-free memory ordering is possible.

Bank conflict avoidance is a graph colouring problem, where the number of permitted colours is determined by the number of banks. For multivariate orthotope B-splines, the B-coefficients are the vertices of the graph and all coplanar coefficients on axis-aligned hyperplanes are connected by edges. Mathematically, an axis-aligned hyperplane is any hyperplane that changes in at most $n - 1$ directions. The problem is similar in some respects to the sudoku graph colouring problem with an additional

[5] This is due to the write-through cache policy.

Table 1 The number of coefficients that are contained within a vertex for different continuities and numbers of dimensions. The amount of shared memory required is the entry multiplied by the number of bytes for the data type. A 7th dimensional orthotope with first-order continuity requires 64 KiB of shared memory when using single-precision floating points and would just fit in shared memory.

Continuity \ Dimensions	1	2	3	4	5	6	7	8
0	2	4	8	16	32	64	128	256
1	4	16	64	256	1024	4096	16384	65536
2	6	36	216	1296	7776	46656	279936	1679616
3	8	64	512	4096	32768	262144	2097152	16777216

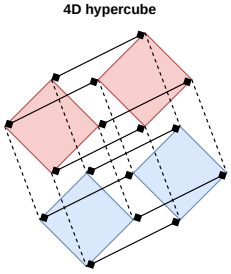


Fig. 28 A 4D hypercube can be interpreted as two 3D hypercubes (cubes) connected at the same vertices. The 3D hypercubes can then be decomposed into a pair of 2D hypercubes (squares).

relaxation that colours may reoccur within a hyperplane provided no bank conflict occurs (proper colouring). This also relaxes the requirement of connecting all vertices on the same hyperplane to a subset of the vertices, where the number of connected vertices is determined by the number of banks available. Alternatively, when this is not possible, the number of bank conflicts should be minimised (improper colouring), which is an optimisation problem in itself. Moreover, the colours (or indices) should be computationally-efficient to compute to mitigate the cost of bank conflicts. First-order continuity for a trivariate regression problem requires $3^3 = 27$ threads and $3^3 + 3^2 = 36$ banks for a simple algorithm for conflict-free access to a block of data (see Fig. 29). This leaves 5 threads idle but results in more constraints being computed per iteration. Higher than first-order continuity with B-coefficient deduplication cannot be efficiently solved with this approach as it does not generalise to higher continuities without requiring a significant increase in the number of memory banks. Instead, without B-coefficient deduplication, it is possible to create a (2,2,2)-cube of B-coefficients that only requires 8 banks. This conflict-free block can then be tiled in higher dimensions guaranteeing conflict-free access for all out-of-edge dimensions. Additionally, since 8 banks are required for the cube, all 32 banks and threads can be used when 8 cubes are available - which is the case for first-order continuity in 4 dimensions.

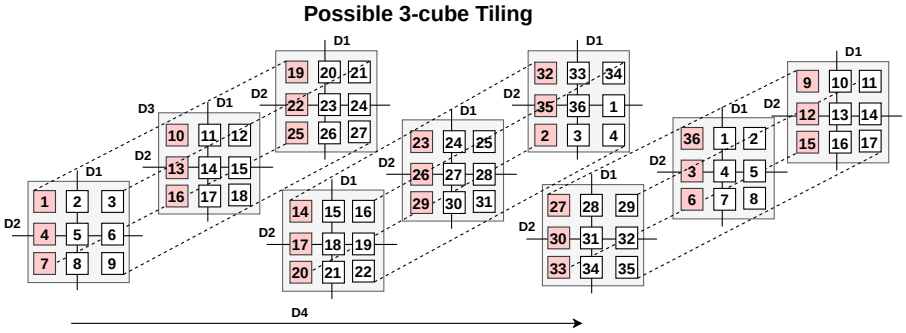


Fig. 29 One plausible conflict-free tiling that can be used for (3,3,3)-cube; however, it requires 36 memory banks and Nvidia GPUs only have 32. Additionally, only 27 threads will typically be active instead of the maximum capacity of 32, although this fact is outweighed by reducing redundant computation by using deduplication (see Appendix B). The amount of memory padding required can be mitigated using modulo addressing, as each address in a (D1,D2,D3)-cube is only used once.

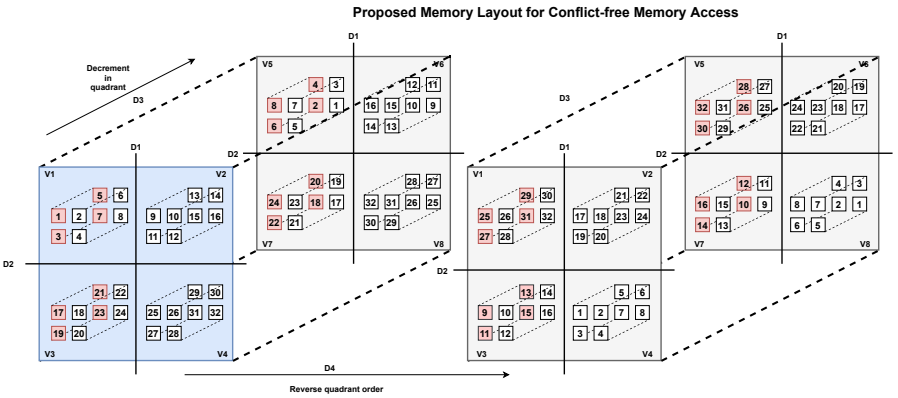


Fig. 30 This figure illustrates which B-coefficients needs to be stored in each memory bank and the order in which they should be read or written to ensure that accesses are conflict-free when performing Kaczmarz's algorithm. The algorithm distinguishes access patterns across constraints in the directions D1, D2, and D3 to those in D4 and higher. For D4 and higher, the B-coefficients enclosed in a square (e.g. blue square) can be assigned to a single warp. For D1, the B-coefficients read and written to by a single warp is coloured in red. The approaches for D2 and D3 follow analogously from D1. Notice that the vertex number still indicate V1-V8 when moving in the D4 direction, this is because D4 is four copies of the (D1,D2,D3)-cube wide and only two copies have been shown for readability.

8 Conclusions & Recommendations

Orthotope B-splines are better for high-dimensional regression as the algorithms required to implement them are simpler and generalise better than those for high-dimensional simplex splines. In this paper, oblique projections and first-order optimisation were demonstrated to improve results by avoiding overfitting - which is often the case in high-dimensional space - and avoiding constraints to be applied between splines when one spline is missing data, as it reduced the training and validation accuracy. Moreover, orthotope B-splines with a different memory organisation has

the potential to significantly improve performance, on both a CPU and a GPU as the cache hierarchy and memory bandwidth can be better utilised. Additionally, the (2,2,2)-cube tiling is compact and results in no padding of shared memory. Finally, a conflict-free use of shared memory on the GPU is critical to enable the scalability and parallelisability of the algorithm.

Further research should investigate using directional derivatives to improve the model in regions where insufficient data is available - while avoiding overfitting. Additionally, investigating how this influences model accuracy in the online setting. It was noted the first projection direction biases the constrained solution to favour the initial projection directions over the latter directions. Research into a weighted solution that minimises this bias should be evaluated to see if it influences model accuracy. A CUDA implementation can be useful to run larger models and enable online experiments to be performed to evaluate the algorithm in situ. The current algorithm is primarily suited for first-order continuity. Further research could look at memory tilings for higher-order continuity, which may be beneficial in other fields like structural analysis. Finally, hybridised optimisation approaches, where the Kaczmarz method is used as a smoothing filter to ensure constraints are satisfied if, for example, dual ascent is terminated before convergence of the dual function.

References

1. van den Aarssen, M.: Distributed approach for aerodynamic model identification of the ice aircraft. Master's thesis, Delft University of Technology (2018). URL http://resolver.tudelft.nl/uuid:df32613c-5f3c-484e-84ea-99672939d6a5
2. Awanou, G., Lai, M.J., Wenston, P.: The multivariate spline method for scattered data fitting and numerical solutions of partial differential equations. In: G. Chen, M.J. Lai (eds.) Wavelets and Splines: Athens 2005, pp. 24–75. Nashboro Press (2005)
3. Bertsekas, D.P.: Convex Optimization Algorithms, first edn. Optimization and Computation Series. Athena Scientific (2015)
4. Bin Zubair, H.: Efficient multigrid methods based on improved coarse grid correction techniques. Ph.D. thesis, Delft University of Technology (2009). URL http://resolver.tudelft.nl/uuid:e0b43b38-ad07-4076-baf1-aa02579c397f
5. Boeing: Statistical summary of commercial jet airplane accidents: Worldwide operations 1959-2018. Online (2019). 50th Edition
6. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning **3**(1), 1–122 (2010)
7. Chen, C., He, B., Ye, Y., Yuan, X.: The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. Mathematical Programming **155**(1-2), 57–79 (2014). DOI 10.1007/s10107-014-0826-5
8. de Visser, C.C., Brunner, E., Verhaegen, M.: On distributed wavefront reconstruction for large-scale adaptive optics systems. Journal of the Optical Society of America A **33**(5), 817–831 (2016). DOI 10.1364/JOSAA.33.000817
9. de Visser, C.C., Verhaegen, M.: Wavefront reconstruction in adaptive optics systems using nonlinear multivariate splines. Journal of the Optical Society of America A **30**(1), 82–95 (2013). DOI 10.1364/josaa.30.000082
10. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research **12**, 2121–2159 (2011)
11. Dykstra, R.L.: An algorithm for restricted least squares regression. Journal of the American Statistical Association **78**(384), 837–842 (1983). DOI 10.2307/2288193
12. Dykstra, R.L., Boyle, J.P.: An algorithm for least squares projections onto the intersection of translated, convex cones. Journal of Statistical Planning and Inference **15**, 391–399 (1986-1987). DOI 10.1016/0378-3758(86)90111-4

13. European Aviation Safety Agency: Executive director decision 2015/012/r. Online (2015)
14. European Aviation Safety Agency: Explanatory note to decision 2015/012/r. Online (2015)
15. Federal Aviation Administration: Qualification, service, and use of crewmembers and aircraft dispatchers. Online (2013). URL https://www.govinfo.gov/content/pkg/FR-2013-11-12/pdf/2013-26845.pdf
16. Gaffke, N., Mathar, R.: A cyclic projection algorithm via duality. Metrika **36**, 29–54 (1989). DOI 10.1007/BF02614077
17. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org
18. Hooij, B.: Online distributed approach for aerodynamic model identification of the ice aircraft. Master's thesis, Delft University of Technology (2020). URL http://resolver.tudelft.nl/uuid:a92817a5-b7f9-4d5f-84de-30f6bd78f668
19. Huisman, F.J.: Full flight envelope aerodynamic modelling of the cessna citation ii using physical splines. Master's thesis, Delft University of Technology (2017). URL http://resolver.tudelft.nl/uuid:f9926c24-e86b-4eee-8986-d427fb4bc667
20. International Air Transport Association: Guidance Material and Best Practices for the Implementation of Upset Prevention and Recovery Training, second edn. International Air Transport Association (2018)
21. International Air Transport Association: Loss of Control In-Flight Accident Analysis Report. International Air Transport Association (2019)
22. Jacklin, S.A.: Closing the certification gaps in adaptive flight control software. In: AIAA Guidance, Navigation and Control Conference and Exhibit. American Institute of Aeronautics and Astronautics (2008). URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20090026333.pdf
23. Kaczmarz, S.: Angenäeherte auflösung von systemen linearer gleichungen. Bull. Int. Acad. Polon. Sci. A. pp. 355–357 (1937)
24. Karagöz, R.: Tensor network b-splines for high-dimensional function approximation. Master's thesis, Delft University of Technology (2020). URL http://resolver.tudelft.nl/uuid:d637a2f4-512d-4330-83ae-29f7ef2958ed
25. Keskar, N.S., Socher, R.: Improving generalization performance by switching from adam to sgd. Tech. rep., Salesforce Research (2017). URL https://arxiv.org/abs/1712.07628.pdf
26. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference for Learning Representations (2015). URL https://arxiv.org/abs/1412.6980
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
28. Lévy, J., Ren, S., Mushtaq, H., Bertels, K., Al-Ars, Z.: GASAL2: a GPU accelerated sequence alignment library for high-throughput NGS data. BMC Bioinformatics (2019). DOI https://doi.org/10.1186/s12859-019-3086-9
29. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of Learning and Motivation **24**, 109–165 (1989). DOI 10.1016/S0079-7421(08)60536-8
30. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**, 529533 (2015). DOI 10.1038/nature14236
31. Nesterov, Y.: A method for unconstrained convex minimization problem with the rate of convergence O(1/k2). Doklady AN USSR **269**, 543–547 (1983)
32. Shewchuk, J.R.: An introduction to the conjugate gradient method without the agonizing pain. Tech. Rep. 1.25, Carnegie Mellon University (1994)
33. Smaragdos, G., Chatzikonstantis, G., Kukreja, R., Sidiropoulos, H., Rodopoulos, D., Sourdis, I., Al-Ars, Z., Kachris, C., Soudris, D., Zeeuw, C.I.D.: Brainframe: a node-level heterogeneous accelerator platform for neuron simulations. Journal of Neural Engineering (2017). DOI https://doi.org/10.1088/1741-2552/aa7fc5
34. van der Peijl, I.: Physical splines for aerodynamic modelling of innovative control effectors. Master's thesis, Delft University of Technology (2017). URL http://resolver.tudelft.nl/uuid:7730dd69-2bb8-4047-8f6d-5681cfd04cee
35. de Visser, C.C.: Global nonlinear model identification with multivariate splines. Ph.D. thesis, Delft University of Technology (2011). URL http://resolver.tudelft.nl/uuid:6bc0134a-0715-4829-903d-6479c5735913

36. Visser, T., de Visser, C., van Kampen, E.: Quadrotor system identification using the multivariate multiplex b-spline. In: AIAA Atmospheric Flight Mechanics Conference. AIAA (2015). DOI doi.org/10.2514/6.2015-0747. URL http://resolver.tudelft.nl/uuid:6e50cda5-3c14-415e-b800-3e1dd42102b5
37. Wei, E., Ozdaglar, A.: Distributed alternating direction method of multipliers. In: 2012 IEEE 51st IEEE Conference on Decision and Control, pp. 5445–5450. IEEE (2012). DOI 10.1109/CDC.2012.6425904
38. Wei, E., Ozdaglar, A.: On the O(1/k) convergence of asynchronous distributed alternating direction method of multipliers. In: 2013 IEEE Global Conference on Signal and Information Processing, pp. 551–554. IEEE (2013). DOI 10.1109/GlobalSIP.2013.6736937
39. Zhu, Y., Rong Li, X.: Recursive least squares with linear constraints. Communications in Information and Systems **7**(3), 287–312 (2007). URL https://projecteuclid.org/euclid.cis/1201531976

A B-spline Directional Derivatives

Directional derivatives are useful for analysis, especially in the context of aerodynamic system identification and control. The directional derivatives in physical space can, for example, be used to assess the static stability of the aircraft. They can also be used to design a controller to meet certain requirements or quantify the inability to meet them.

The De Casteljau Matrix from [35] - given in Eq. 31, where each entry is formed using Eq. 30 - is a reformulation of the recursive De Casteljau algorithm into a non-recursive matrix form. $\hat{d}_1$ and $\hat{d}_2$ are given by filling in the first and second superscripts into Eq. 4, respectively.

$$P_\gamma^m(b) = \begin{cases} \frac{m!}{\gamma!} b^m & \text{if } \gamma \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

$$\mathbf{P}^{d,d-m}(b) = \left[P_{\theta-\kappa}^m(b) \right]_{\substack{\theta \in \mathbb{M}_d^{n+1} \\ \kappa \in \mathbb{M}_{d-m}^{n+1}}} \in \mathbb{R}^{\hat{d}_2 \times \hat{d}_1} \tag{31}$$

The directional derivative is taken with respect to a directional vector u. Unfortunately, u is a vector and cannot be converted to barycentric coordinates; however, to overcome this problem, the vector u can be described as the difference between two points in Cartesian space (see Eq. 34) and these points can then be transformed into barycentric space and then subtracted from each other and denoted as a (see Eq. 35) [35]. The m^{th} directional derivative of the polynomial p with respect to the direction u is given by Eq. 36.

$$B_\kappa^d(b) = \sum_{|\gamma|=m} P_\gamma^m(b) B_{\kappa-\gamma}^{d-m}(b) \tag{32}$$

$$p(b) = \mathbf{B}^{d-m}(b) \mathbf{P}^{d,d-m}(b) \cdot c^{t_j} \tag{33}$$

$$u = v - w \in \mathbb{R}^n \tag{34}$$

$$b(u) \coloneqq a = b(v) - b(w) \in \mathbb{R}^{n+1} \tag{35}$$

$$D_u^m p(b) = \frac{d!}{(d-m)!} B^{d-m}(b) \mathbf{P}^{d,d-m}(a) \cdot c^{t_j} \tag{36}$$

B B-coefficient Deduplication

This section has been placed in the appendix as the B-coefficient deduplication does not work well in combination with Section 7.2. This is because the deduplication leads to odd widths for the vertex blocks and the bank conflict avoidance scheme requires tilings of even width blocks due to the limited number of memory banks.

At the edges connecting two splines, the unconstrained B-coefficients represent the same physical-space location but may have different values. After applying zeroth-order continuity across the edge, they are also equal in value. If these duplicated coefficients are not removed, each iteration will require that

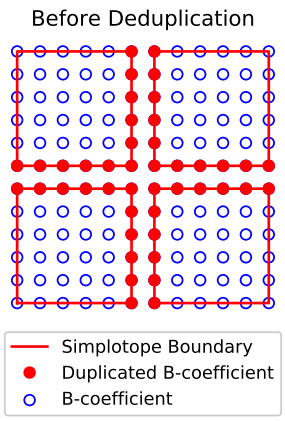


Fig. 31 Splines have been artificially separated to illustrate the duplicated coefficients.

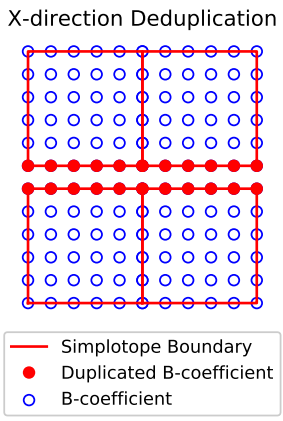


Fig. 32 B-coefficients in the x-direction have been deduplicated by applying zeroth-order continuity in the x-direction. Splines have been artificially separated to illustrate the duplicated coefficients.

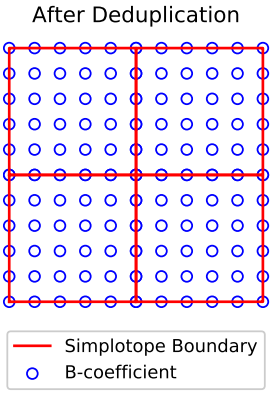


Fig. 33 B-coefficients in the y-direction have been deduplicated by applying zeroth-order continuity in the y-direction. Splines have been artificially separated to illustrate the duplicated coefficients.

zeroth-order continuity be reapplied and lead to slower overall convergence. This is obviated by using orthogonal constraints, as mentioned in Section 6; however, this still requires additional memory store operations and multiplications - if not deduplicated. Deduplication improves scalability as base of the exponential growth in higher dimensions is reduced by one, as given by Eq. 37. For first-order continuity in the 3D case, 2.37 times fewer memory is required and the benefit grows with more dimensions. The amount of redundant computation is significantly reduced as one fewer multiplication is required per projection and the number of required projections decreases from $(2 \cdot \hat{r})^{n-1}$ to $(2 \cdot \hat{r} - 1)^{n-1}$. A lower memory load means more data can be fit into the cache or, in the case of CUDA, shared memory. The deduplication is performed using twice the duplicated coefficient's value in Eq. 25 and updating the coefficient using Eq. 26 with half its value.

$$\mathbf{M} = \frac{(2 \cdot \hat{r})^n}{(2 \cdot \hat{r} - 1)^n} \tag{37}$$

When utilising oblique projections, the B-coefficient deduplication process needs to be weighted based on the number of splines with useful data. The weighting is changed each iteration to be the sum of the weights of the two splines involved for that direction. Each orthotope needs a weight for each direction. The weight for a B-coefficient is determined as the sum of the weights of the directions it is involved with. For example in the 2D case, a vertex of an orthotope is involved in 2 directions and an edge is only involved in one direction.

C Performance Analysis

This section will analyse the speedup of the algorithm for modifications to the storage of data. The algorithms have been implemented in *MATLAB* and the results may differ when optimising the algorithms for other platforms or in other programming languages. The DAB algorithm can operate in two main scenarios: online and batch. When using the algorithm in the online regression setting, typically the computational capabilities is significantly decreased as in the case of micro unmanned aerial vehicles, where the L1 cache size is typically around 8 KB.

The experiments were conducted on a hexacore Ryzen 3600 processor with a clock frequency of 4.2 GHz and 32 GB of RAM at 2933 MHz. The processor has 32 KB of L1 data cache per core and 32 MB

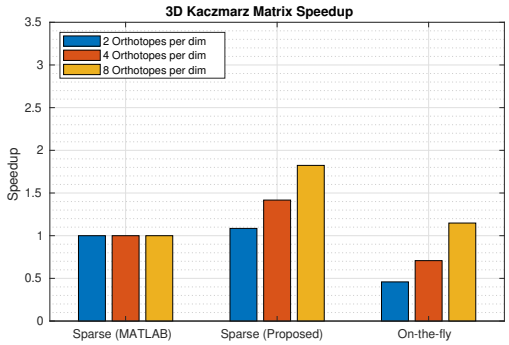


Fig. 34 The speedup over using the *MATLAB* sparse matrix format for the constraint matrix H in the 3D case with different resolution grids. The benefit of on-the-fly and the proposed sparse format improves as the grid becomes finer.

of shared L3 cache. The data for approximately 500 single-precision floating-point 7D orthotopes could in theory fit in the L3 cache. Of course, in practice the number of orthotopes will be slightly less as no fine-grained control of the cache is available. Additionally, if a sparse matrix format is used - in contrast to an on-the-fly approach - this data will also consume cache space. L3 cache is; however, 10 times slower than L1 cache with respect to latency on the selected system. Execution times are averaged over 10 runs and the code tested is, specifically, the orthogonal Kaczmarz projection algorithm with orthogonal constraints.

One optimisation applied to the DAB algorithm was to reduce the total amount of memory required. The constraint matrix is very sparse. Using a sparse matrix format requires storing the indices and the values of each non-zero entry. The values are constant for a given continuity and can be stored separately from the indexing. The indexing data can be stored in an array with a column for each non-zero entry in a row and a row for each constraint. The columns of the Index array maps one-to-one with the columns of the Value array. There is one Value and one Index array per continuity order and per continuity direction as the dimensions of the arrays are dependent on the continuity order and each direction is handled separately. The proposed approach is similar to the Compressed Sparse Row (CSR) format except with a set of values per continuity order instead of per non-zero entry and the row pointer is implicit as all rows have the same width. Finally, a comparison will be performed with on-the-fly computation of the indices, as computing the indices and the neighbouring orthotopes is trivial with the regularity of the tessellation. It should be noted that *MATLAB* uses the Compressed Sparse Column (CSC) format for storing sparse matrices. In Fig. 34, the MATLAB sparse storage format outperforms the on-the-fly approach for small tessellations; however, as the size of the tessellation increases, so does the speedup of the on-the-fly approach. From Fig. 34, Fig. 35, and Fig. 36, it can be seen that the proposed sparse format always outperforms the sparse method implemented by *MATLAB*. With increased dimensions and tessellation sizes, the proposed sparse format is the best up to at least the 5D case, as tested; however, a slight decrease in performance is observed in Fig. 35 and Fig. 36, as the L3 cache is filled beyond its capacity. The benefit of the proposed sparse method is also more aimed at CPU computation, as GPUs are often memory bottlenecked and the larger number of cores are better suited to on-the-fly computation. The execution times are provided in Table 2. It should be noted that all implementations utilised the optimisation that the denominator of the projection is the same for all projections for a given continuity order and does not require an expensive dot product.

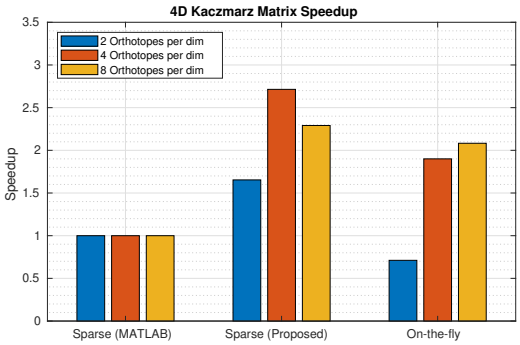


Fig. 35 The speedup over using the *MATLAB* sparse matrix format for the constraint matrix H in the 4D case with different resolution grids. The proposed sparse method has a reduced speedup for the case of 8 orthotopes per dimension because the limits of the L3 cache have been reached. The on-the-fly approach still improves as computing the values is faster than loading them from main memory.

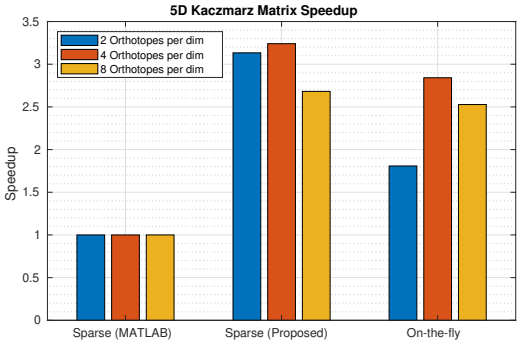


Fig. 36 The speedup over using the *MATLAB* sparse matrix format for the constraint matrix H in the 5D case with different resolution grids. The on-the-fly approach also begins to lose performance as the computed indices are being temporarily sent to main memory. Decomposing the on-the-fly computation into batches may further improve the performance.

Table 2 This table shows the average execution times in seconds for different tessellation sizes using the different approaches. In the 5D case with 8 orthotopes per dimension, the execution time takes around 5.2 seconds to complete with the fastest method.

	Sparse (MATLAB)			Sparse (Proposed)			On-the-fly		
Dimensions	2/dim	4/dim	8/dim	2/dim	4/dim	8/dim	2/dim	4/dim	8/dim
3D	3.05E-04	5.74E-04	0.0031	2.81E-04	4.05E-04	0.0017	6.65E-04	8.11E-04	0.0027
4D	6.68E-04	0.0076	0.1787	4.04E-04	0.0028	0.078	9.39E-04	0.004	0.0858
5D	0.0047	0.2492	14.0383	0.0015	0.0769	5.2348	0.0026	0.0877	5.5535

II

Literature Survey

14

Function Approximation

This chapter aims to clarify why function approximation is necessary and elaborate on the trade-offs between the various types of function approximators. Finally, it will provide a detailed background on the multivariate simplex and simplotope B-splines and provide details on some useful properties of them, particularly those with application to aerodynamic system identification.

14.1. Regression vs Interpolation

The two main approaches to approximate data points that are not contained within the dataset are: interpolation and regression. Interpolation requires that the approximated function passes through the provided data points whereas regression does not impose this constraint.

Interpolation is a simple approach; however, the amount of storage required to store all data points and the data structures required to efficiently access them often make them infeasible for large datasets. Moreover, experimental data is typically not noise-free and interpolating the data would model the noise and not the actual process being modelled. A consequence of modelling the noise would mean that the derivative information obtained from the model would be meaningless for understanding the underlying process. The derivative information is also important in the context of aerodynamic model identification as this is used in determining the aerodynamic stability of the aircraft.

Regression, in contrast, imposes a cost function that needs to be minimised to provide the best fit. Weierstrass's approximation theorem indicates that any function can be approximated to an arbitrary precision. However, regression requires knowledge of the degree of the underlying process being modelled to avoid modelling the error, which would reduce the cost on the training dataset, and to avoid underfitting the model. Some approaches to avoid overfitting are to penalise large parameters in the cost function, see section 15.2. To avoid having to employ automatic processes to determine the optimal degree in online systems, it is useful to subdivide the domain and apply local models of lower degree. This approach is known as spline regression. This has additional benefits for avoiding numerical precision issues associated with matrices with large magnitude differences in its elements (see Section 15.5.1) and to avoid problems such as Runge's Phenomenon for modelling highly nonlinear processes. The use of local models means the optimisation can also be performed locally and in parallel; however, in order to avoid discontinuities between the splines, it is necessary to apply constraints which complicates the optimisation process by coupling the local models.

Weierstrass Approximation Theorem

Any continuous function on a closed and finite interval [a,b] can be approximated by a polynomial function within some desired tolerance.

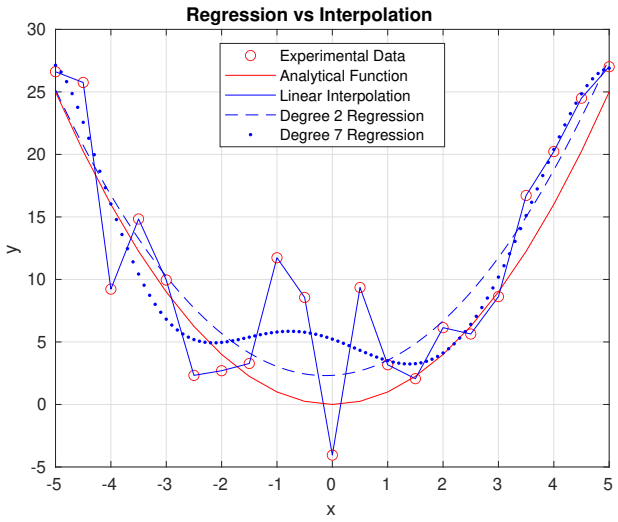


Figure 14.1: The analytical function is sampled at discrete points and corrupted with Gaussian noise. Interpolation requires the function approximator to pass through the data points and regression tries to fit the data as closely as possible without modelling the noise in the experimental data. The degree 7 regressor has a lower error but is a worse fit to the analytical function.

Runge's Phenomenon

A polynomial approximating a continuous function on a closed interval [a,b] oscillates towards the boundaries of the interval.

14.2. Overview of Function Approximation Methods

This section will review some common approaches to function approximation using regression.

Fuzzy-blending is a manual approach to smoothing and connecting splines that requires expert knowledge [14]. The smoothing must also be reperformed if the granularity of the grid is changed. Moreover, higher-order continuity cannot be achieved, which is of importance for computing the control effectiveness Jacobian.

Tensor-product splines requires taking the tensor-product of 1D polynomials. The downside of these tensor-products is that the tensor-product can create precision errors when working in higher dimensions. But most importantly, the data needs to be collected on a rectangular grid [14] and for some applications, such as aerodynamic flight data where data is scattered, this is not possible nor desirable.

Thin-plate splines is another approach that can, in contrast to tensor-product splines, handle scattered data. The downside to thin-plate splines is that it requires a global radial basis function for each data point [14]. As a result of the global nature of the basis functions, all basis functions are required to evaluate and optimise the spline, which can quickly become computationally infeasible. This is exacerbated as the dataset increases in size.

Artificial Neural Networks (ANN) are a set of algorithms that are modelled loosely on the neural system. These algorithms are among the most popular in recent times due to the power and potential these algorithms have shown in domains such as reinforcement learning [47] and computer vision[40], especially when it comes to generalisation. The benefit of generalisation (extrapolation) may be of limited value for highly nonlinear aircraft and should be investigated separately. The idea is to combine a linear function with a nonlinear function in a series of layers. The most common training algorithm for ANNs is backpropagation and involves computing the gradients at each point within the network and determining its influence on the output. The output is then adapted to make the output match

the training samples and is typically optimised with various adaptations of gradient descent. ANNs are powerful and can approximate any nonlinear function provided enough layers and neurons are present according to the universal approximation theorem. The major problems with ANNs are that they are non-convex, black-box systems, computationally dense, and require modifications to ensure that continuity and smoothness is satisfied. By computationally dense, it is meant that even if the outputs of the network are sparse, the entire network needs to be evaluated to determine the output. Spiking Neural Networks are a new generation of neural networks for neuromorphic hardware that will be able to exploit this sparsity. Moreover, since they are black-box systems, the entire ANN needs to evaluated multiple times to estimate derivative information which can be computationally expensive depending on the size of the network. Furthermore, small changes in the input do not necessarily mean small changes in the output as a result of the nonlinearities. It is also much harder to ensure all constraints are satisfied throughout the entire domain as they suffer from problems such as catastrophic forgetting (see [46] for more details). Catastrophic forgetting is also more apparent for online learning as the data is highly correlated. For example in reinforcement learning, nonlinear function approximators, like neural networks, are unstable as a result of this and have required modifications like memory buffers of past data and copies of the networks to stabilise the algorithms [47]. The non-convexity of ANNs lead to slower training, as higher-order optimisation algorithms may converge to a saddle point or maximum. Although, there exist modifications that solve this problem, see for example [12]. Additionally, higher-order algorithms will be unstable for locally linear functions as the Hessian will not be invertible. First-order algorithms can only guarantee convergence to a local minima and can be very slow; however, first-order algorithms and local minima are often sufficient in practice. The size of the networks and the plethora of hyper-parameters that are associated with them are empirically determined and provide no guarantees of their performance in-situ and complicates the design and certification process [34]. Finally, as a result of ANNs being nonlinear mapping operators, they are able to avoid the curse-of-dimensionality to an extent as sparse high-dimensional data can be mapped to a more dense memory-efficient lower-dimensional representation.

Multivariate Simplex and Simplotope B-Splines are linear-in-the-parameters function approximators that can be used to model nonlinear data. They can be solved using Linear Least Squares and can therefore be optimised using convex solvers. Since it is a linear-in-the-parameters function approximator, there are more guarantees for the stability of the approximator and the mathematical tools for analysis of the approximator are more mature. The curse-of-dimensionality is present since no nonlinear map is used. It is, nevertheless, possible to use dimensionality reduction techniques, such as autoencoders [27], principle component analysis, tensor networks [36], and hashing functions to alleviate this but, as with any compression technique, some data loss will occur and accuracy will be decreased. The extent to which this occurs can be managed with appropriate trade-offs. Nonetheless, the interpretability is lost, if a dimensionality reduction technique is used. Moreover, it should remain static to avoid invalidating the function approximator that relies on that input representation. The curse-of-dimensionality is additionally abated by the use of local basis functions, as these result in a very sparse problem structure and can be efficiently evaluated. The optimisation of the parameters can also occur in a distributed and parallel fashion.

14.2.1. Multivariate Simplex B-splines

A simplex t_j is a geometrical structure that minimally spans a set of n dimensions and is composed of exactly $n+1$ vertices where the edges are given by simplices of one dimension lower. Simplices can be connected to each other at their edges. A collection of these connected simplices is called a triangulation $\mathcal{T}$.

Each simplex has a local coordinate system known as the barycentric coordinate system. A useful property of barycentric coordinates is that points that lie outside the simplex have one or more negative barycentric coordinates. To convert Cartesian coordinates to barycentric coordinates, the transformation matrix $T \in \mathbb{R}^{n\times n}$ for a simplex should be formed. The barycentric coordinates $b \in \mathbb{R}^{n+1}$ can then be computed using Eq. 14.2 and Eq. 14.3. The vertices v of the simplex are numbered using the superscript and their Cartesian coordinates are indexed by their subscript.

$$T = \begin{bmatrix} v_0^1 - v_0^0 & \cdots & v_0^n - v_0^0 \\ \vdots & \vdots & \vdots \\ v_{n-1}^1 - v_{n-1}^0 & \cdots & v_{n-1}^n - v_{n-1}^0 \end{bmatrix} \tag{14.1}$$

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = T^{-1} \begin{bmatrix} x_0 - v_0^0 \\ x_1 - v_1^0 \\ \vdots \\ x_{n-1} - v_{n-1}^0 \end{bmatrix} \tag{14.2}$$

$$b_0 = 1 - b_1 - b_2 - \cdots - b_n \tag{14.3}$$

The Bernstein basis functions are given by the multinomial theorem. It is a stable basis that sums to 1. Additionally, it is local because the basis functions are zero outside of the simplex. Each basis function B_κ^d is scaled by its B-coefficient c_κ to approximate any polynomial, as given by De Boor's theorem. $\mathbf{B}^d \in \mathbb{R}^{m \times \hat{d}}$ is the matrix formulation of the summation in Eq. 14.7, where m is given by the number of barycentric coordinates being used and $\hat{d}$ is given by Eq. 14.4. In the case of the theorem, $m = 1$. $\vec{1}$ is a column vector of ones. The B-coefficients each belong to a specific spline t_j and may be distinguished by a superscript to indicate this membership. The barycentric coordinate of each B-coefficient with respect to a spline of degree d is given by Eq. 14.10. $\kappa \in \mathbb{N}_0^{n+1}$ is a multi-index. An additional notation used for multi-indices is $\mathbb{M}_d^{n+1}$, where the superscript denotes the set of $\mathbb{N}_0^{n+1}$ subject to the constraint that the norm of the multi-index is equal to the subscript d and the number of elements in the set is determined by filling in the subscript in Eq. 14.4.

$$\hat{d} = \frac{(d+n)!}{d!n!} \tag{14.4}$$

Multinomial Theorem for Barycentric Coordinates

$$(b_0 + b_1 + \cdots + b_n)^d = \sum_{|\kappa|=d} \frac{d!}{\kappa_0!\kappa_1! \cdots \kappa_n!} b_0^{\kappa_0} b_1^{\kappa_1} \cdots b_n^{\kappa_n} \tag{14.5}$$

$$= \sum_{|\kappa|=d} \frac{d!}{\kappa!} b^\kappa \tag{14.6}$$

$$= \sum_{|\kappa|=d} B_\kappa^d(b) \tag{14.7}$$

$$= \mathbf{B}^d \vec{1} \tag{14.8}$$

$$= 1 \tag{14.9}$$

De Boor's Theorem

Any polynomial $p(x)$ of degree d can be written as a sum of basis functions B_κ^d scaled by its B-coefficient $c_\kappa^{t_j}$:

$$p(x) = \sum_{|\kappa|=d} c_\kappa^{t_j} B_\kappa^d(b(x))$$

$$b(c_\kappa) = \frac{\kappa}{d} \tag{14.10}$$

Since the basis functions are local to a simplex, the evaluation of the multivariate simplex B-spline for any point only requires one to evaluate the function for the simplex that contains the point. During

regression, this manifests itself as a sparse block-diagonal structure which significantly reduces the computational complexity, especially for large triangulations.

Algorithm 1: Regression with Multivariate Simplex B-splines

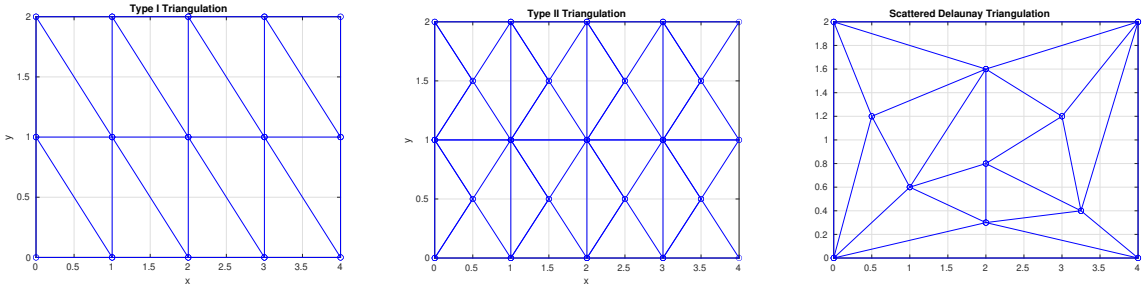
1 Determine data domain.
2 Create triangulation $\mathcal{T}$ for the domain.
3 Generate constraint matrix for $\mathcal{T}$ of degree d and continuity $\mathbb{C}^r$.
4 **while** *online* **do**
5 Determine simplex membership of data point in $\mathcal{T}$ and convert to barycentric coordinates.
6 Update basis function regression matrix.
7 Use equality-constrained least squares to determine $c_k^{t_j}$.
8 **end**

See De Visser's PhD thesis [14] for more detailed information on multivariate simplex B-splines.

14.2.2. Triangulation

As mentioned before, a tessellation of simplices is called a triangulation. Type I and Type II triangulations are created with vertices that are placed on a grid and are the simplest triangulations. More complex triangulations can, for instance, be generated using Delaunay triangulation, although care should be taken as the quality of the simplices may introduce numerical issues due to poor data distribution and can reduce the approximation power of a given simplex for certain directions.[14]

Even though many different valid triangulations exist for a given domain, some tessellations may be better than others for certain purposes as De Visser demonstrated in [14]. For example, a type I triangulation has fewer neighbours and total simplices for a given domain when compared with an equivalent type II triangulation. In contrast, De Visser observed a type II triangulation is more favourable for damping geometric constraint propagation. Coxeter-Freudenthal-Kuhn triangulations (or Kuhn triangulations) are a generalisation of type I (Freudenthal's triangulation) and type II (Tucker's triangulation) triangulations to higher-dimensional spaces [48].



(a) (b) (c)

Figure 14.2: Three valid triangulations for the same domain.

14.2.3. Constraints

There are two main types of constraints that are applicable to aerodynamic function approximation. The first being continuity constraints between the simplices in the triangulation and the second being a priori knowledge of the problem.

Continuity constraints are particularly important for control systems that are based on the estimated aerodynamic model as discontinuities or also lack of differentiability between simplices will either limit the control algorithms that can be applied or limit the performance of the control system. The discon-

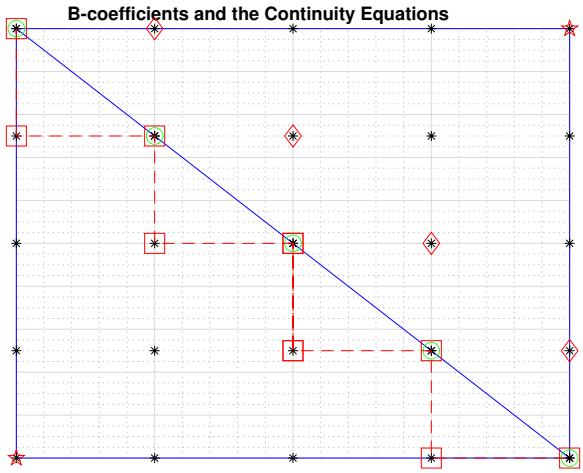


Figure 14.3: Asterisks indicate the spatial locations of B-coefficients relative to each fourth order simplex according to Eq. 14.10. The stars indicate the out-of-edge vertex. Zeroth order continuity is given by the 5 green circles. For the first order continuity, the squares and diamonds are involved in the first-order continuity. The dashed red line indicates adjacent B-coefficients that are involved in the same continuity equation. Note, first-order continuity implies zeroth-order continuity.

tinuities and lack of differentiability is also an artifact of using splines and not necessarily related to the underlying function that is being approximated. $v_*^{t_i}$ is the barycentric coordinate of the out-of-edge vertex for simplex t_i with respect to simplex t_j. The non-zero multi-index for the out-of-edge vertex for simplex t_i is denoted as $\kappa_*^{t_i}$. Finally, γ is a multi-index. It should be noted that the continuity matrix is purely a function of the triangulation and its connectivity and is not dependent on the data being regressed. This means that this matrix is always constant. A consequence of applying the spline continuity constraints is that the local model becomes a global model. The distance the constraints propagate through the model is related to the number of degrees of freedom of each simplex [14]. Therefore, higher-order models with lower-order continuity leads to more local models. The upper-bound on the number of constraints is given by Eq. 14.15 multiplied by the number of shared edges. This is also the number of constraints generated by following the algorithm in Eq. 14.11 and, as a result, some constraints are redundant which has repercussions for some solvers (see for example Eq. 15.32). If the nullity of the constraint matrix is zero, only the trivial solution exists.

Continuity Equations

$$c_{\kappa_0,\cdots,\kappa_n}^{t_i} = \sum_{|\gamma|=m} c_{(\kappa_0,\cdots,\kappa_n)+\gamma}^{t_j} B_\gamma^m(v_*^{t_i}), \qquad \{\forall m \in \mathbb{N}_0 | 0 \leq m \leq r\} \tag{14.11}$$

subject to:

$$\kappa_0^{t_i} + \cdots + \kappa_n^{t_i} = d \tag{14.12}$$

$$\kappa_*^{t_i} = m \tag{14.13}$$

$$\kappa_*^{t_j} = 0 \tag{14.14}$$

$$\hat{r} = \sum_{m=0}^{r} \frac{(d-m+n-1)!}{(d-m)!(n-1)!} \tag{14.15}$$

A priori knowledge can for example be incorporated into the function approximation problem by including constraints due to physical constraints associated with the aircraft. However, the simplex B-splines operate using barycentric coordinates and, in order to be able to apply these constraints, the physical constraints in Cartesian coordinates need to be transformed to barycentric constraints. This approach to constraint application is referred to as physical splines in literature [31][63]. It is also possible to

apply constraints based on the directional derivatives as in [15] using the De Casteljau matrix in Section 14.2.4.

14.2.4. Directional Derivatives

Directional derivatives are useful for analysis, especially in the context of aerodynamic system identification and control. The directional derivatives in physical space can, for example, be used to assess the static stability of the aircraft. They can also be used to design a controller to meet certain requirements or quantify the inability to meet them.

The De Casteljau Matrix from [14] given in Eq. 14.17, where each entry is formed using Eq. 14.16, is a reformulation of the recursive De Casteljau algorithm into a non-recursive matrix form. $\hat{d}_1$ and $\hat{d}_2$ are given by filling in the first and second superscripts into Eq. 14.4, respectively.

$$P_\gamma^m(b) = \begin{cases} \frac{m!}{\gamma!} b^m & \text{if } \gamma \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{14.16}$$

$$\mathbf{P}^{d,d-m}(b) = \left[P_{\theta-\kappa}^m(b)\right]_{\substack{\theta \in \mathbb{M}_d^{n+1} \\ \kappa \in \mathbb{M}_{d-m}^{n+1}}} \in \mathbb{R}^{\hat{d}_2 \times \hat{d}_1} \tag{14.17}$$

The directional derivative is taken with respect to a directional vector u. Unfortunately, u is a vector and cannot be converted to barycentric coordinates; however, to overcome this problem, the vector u can be described as the difference between two points in Cartesian space (see Eq. 14.20) and these points can then be transformed into barycentric space and then subtracted from each other and denoted as a (see Eq. 14.21)[14]. The m^{th} directional derivative of the polynomial p with respect to the direction u is given by Eq. 14.22.

$$B_\kappa^d(b) = \sum_{|\gamma|=m} P_\gamma^m(b) B_{\kappa-\gamma}^{d-m}(b) \tag{14.18}$$

$$p(b) = \mathbf{B}^{d-m}(b)\mathbf{P}^{d,d-m}(b) \cdot c^{t_j} \tag{14.19}$$

$$u = v - w \in \mathbb{R}^n \tag{14.20}$$

$$b(u) \coloneqq a = b(v) - b(w) \in \mathbb{R}^{n+1} \tag{14.21}$$

$$D_u^m p(b) = \frac{d!}{(d-m)!} B^{d-m}(b)\mathbf{P}^{d,d-m}(a) \cdot c^{t_j} \tag{14.22}$$

14.3. Multivariate Simplotope B-Splines

Simplotope splines are a generalisation of simplex B-splines that enable the dimensions to be decoupled. Effectively, simplotope splines are tensor-product splines applied to simplices in lower dimensions. Simplex splines are therefore simplotope splines with one layer. In literature, simplotope B-splines are also called multivariate B-splines[36] and multiplex B-splines[64]. These lower dimensional simplices are called layers and each layer is orthogonal to all the other layers. Simplotope equations are denoted with π to distinguish them from their simplex counterparts. The multi-index κ is similar to the those for simplices; however, instead of one multi-index, there is one for each layer which are concatenated to form κ (i.e. $\kappa = [\kappa_1, \cdots, \kappa_l]$). The same applies to the barycentric coordinates b. Simplotope splines have a higher approximation power for the same number of coefficients due to the decoupling of the dimensions. Decoupling the dimensions has an additional benefit when applying physical constraints as the mapping can be made one-to-one. Moreover, using low-dimensional simplices will, depending on the regularity of the triangulation, create common repeating structures in the matrices that can be

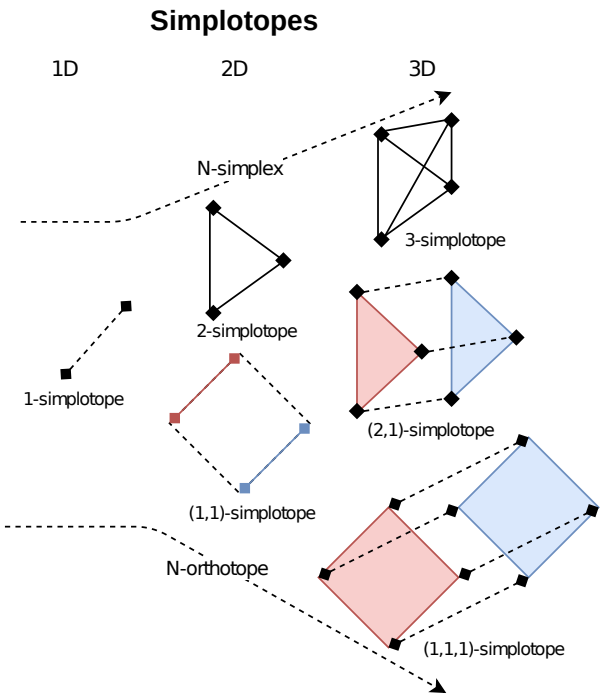


Figure 14.4: Simplotopes can be described as N-simplices and N-orthotopes at the extremes. The figure is a 2D projection of the N-dimensional simplotopes. N-simplices are generated by adding a vertex in the n^{th} dimension. N-orthotopes are created by duplicating the (N-1)-orthotope in the n^{th} dimension and connecting the respective vertices or, alternatively, as a tensor-product of 1D-simplices. Adapted from [61].

exploited to reduce memory and computation requirements and enable techniques like lookup tables to be used.

In addition, generating triangulations for high-dimensional simplotopes consisting of low-dimensional simplices is much easier than the equivalent problem for simplex splines. The algorithm for simplotopes consisting of 1D simplices is trivial and could potentially be used in an online setting. This reduces the size of the domain to optimise as a tensor-product grid will grow rapidly but be sparsely populated. A sparsely populated grid means that in the majority of the domain extrapolation is occurring, which is often not desirable.

The more layers in a simplotope spline, the fewer the number of constraints for the same grid size; however, the number of parameters per simplotope also increases with the number of layers. More parameters per simplotope means larger block sizes in the block diagonal data matrix. The benefit of more layers is that the total number of coefficients are reduced, as fewer coefficients are duplicated and there are fewer simplotopes in the same triangulation and therefore fewer constraints to be applied.

Eq. 14.23 is used to evaluate a simplotope. B_{κ_i} in Eq. 14.24 is the B-form for the simplex in layer i with the subset of the multi-index κ for that layer.

$$\pi(b) = \sum_{\substack{|\kappa_i|=d_i \\ \forall i \in [1,l]}} c_\kappa B_\kappa(b) \tag{14.23}$$

$$B_\kappa(b) = \prod_{i=1}^{l} B_{\kappa_i}^{d_i}(b_i) \tag{14.24}$$

14.3.1. Constraints

The application of constraints is very similar to simplex B-splines; however, requires a slight modification. The process of generating the continuity constraint as in Eq. 14.11 is only applied to the simplex that contains the out-of-edge vertex. The tensor-product is then applied to all permutations of the simplices in the other layers. The number of continuity constraints for an edge of a simplotope is given by Eq. 14.25, where $\hat{r}_*$ is given by Eq. 14.15. $\hat{r}_*$ is the number of constraints for the out-of-edge simplex. $\hat{d}_i$ and $\hat{d}_*$ are the number of parameters in layer i and the out-of-edge simplex, respectively.

$$\hat{r}_\pi = \hat{r}_* \frac{\prod_{i=1}^{l} \hat{d}_i}{\hat{d}_*} \tag{14.25}$$

15

Optimisation

This chapter will deal with the main focus of the literature study: optimisation algorithms. The chapter will begin with an introduction to the theory of convex programming and describe the nature of the optimisation problem for multivariate simplotope B-splines. Next, the various cost functions will be evaluated with a list of pros and cons. Thereafter, separability is an important consideration as it relates to parallelisability of the problem. Derivative-free and gradient-based unconstrained algorithms will then be discussed. Building upon that will be the section on constrained optimisation which sometimes utilises the unconstrained algorithms. Online algorithms are then discussed as these play an important role in adaptive flight control. Finally, multi-grid methods will be discussed.

15.1. Convex Programming

This section will briefly introduce convex programming and the assumptions that are being used for the algorithms presented in the subsequent sections. The focus will be placed on quadratic programming with only equality constraints. This is because the fitting of the multivariate simplotope B-spline model can be formulated as an equality-constrained problem with a quadratic loss function.

A convex function is any function $f(\theta) : \mathbb{R}^n \mapsto \mathbb{R}$ where $\theta \in \mathbb{R}^n$ for which $f''(\theta) \geq 0$ in the case of a smooth convex function. In matrix terminology this equates to the Hessian being positive semi-definite. In the case of a non-smooth convex function, the constraint is given by Eq. 15.1 and states that the subgradient (i.e. vector d) at any point θ_0 is any subgradient that does not violate the constraint. The set of all valid subgradients of f is called the subdifferential set and if such a set exists, the function is subdifferentiable. If the function f is differentiable (i.e. smooth) the inequality becomes an equality and the subgradient is equal to the gradient at the point θ_0. Verbally, the subgradient of a function f at a point θ_0 is the gradient of any tangent hyperplane of the epigraph of f in $\mathbb{R}^{n+1}$ that passes through the point $(\theta, f(\theta)$[3]. A convex function is subdifferentiable for all points $\theta \in \mathbb{R}^n$ or within the specified domain of f. An alternative definition of convexity is given by Jensen's inequality that states a chord between any two points must lie above the graph. A globally optimal solution is any point at which 0 is contained within the subdifferential set.

$$f(\theta) - f(\theta_0) \geq d^T(\theta - \theta_0), \qquad \forall \theta \in \mathbb{R}^n \tag{15.1}$$

$$f(\gamma\theta_0 - (1-\gamma)\theta_1) \leq \gamma f(\theta_0) - (1-\gamma)f(\theta_1), \qquad \{(\theta_0, \theta_1) \in \mathbb{R}^2, \gamma \in [0,1]\} \text{ (Jensen's Inequality - 15.2)}$$

Convex programming (CP) is described by problems of the form given by Eq. 15.3. $f(\theta)$ is some

convex function that is to be optimised and subject to the convex inequality constraints $g_i(\theta)$ and affine equality constraints $h_i(\theta)$. An affine constraint is a linear constraint and a translation and can be succinctly described by $H\theta - d = 0$, where H is a matrix in $\mathbb{R}^{p\times n}$ and d is a vector in $\mathbb{R}^p$. A strongly convex function is one for which the lower-bound on the increase of the convex function is quadratic. This requires that the Hessian is positive definite.

Convex Programming

$$\begin{aligned} \underset{\theta}{\text{minimise}} \quad & f(\theta) \\ \text{subject to} \quad & g_i(\theta) \leq c_i, \quad \forall i \in [1, q] \\ & h_i(\theta) = d_i, \quad \forall i \in [1, p] \end{aligned} \tag{15.3}$$

Convex quadratic programming (CQP) (Eq. 15.4) simplifies the problem by requiring $f(\theta)$ to be at most a quadratic function of θ and $g_i(\theta)$ to also be an affine constraint with the matrix $Q \in \mathbb{R}^{n\times n}$ symmetric and positive semi-definite and $r \in \mathbb{R}^n$. Note that the matrix Q is also the Hessian of the loss function. As mentioned before, the problem that needs to be solved for the optimal constrained parameters of the multivariate simplotope B-spline model is given by Eq. 15.4 where $g_i(\theta) = 0 \quad \forall i \in [1, q]$ unless physical or boundary constraints are applied (see Section 14.2.3). Since convex quadratic programming is a subset of convex programming, any method that applies to CP can be applied to solve it. However, since CQP is simpler and has additional assumptions the problem can typically be solved faster and more efficiently when exploiting these assumptions. Closed-form solutions for CQP do exist; however, these solutions do not typically scale well to higher dimensions or when data coverage is low, which is exacerbated by more dimensions. Low data coverage means the Hessian is not positive definite and the loss function is not strongly convex.

Convex Quadratic Programming

$$\begin{aligned} \underset{\theta}{\text{minimise}} \quad & \tfrac{1}{2}\theta^T Q\theta + r^T\theta \\ \text{subject to} \quad & g_i(\theta) \leq c_i, \quad \forall i \in [1, q] \\ & h_i(\theta) = d_i, \quad \forall i \in [1, p] \\ & Q \succeq 0 \\ & Q = Q^T \end{aligned} \tag{15.4}$$

15.2. Cost Functions

Many different cost functions or loss functions can be used for determining the optimal approximator; however, each cost function has different properties and treats outliers differently. First, the residual r is given by the difference between the function approximator $\mathcal{F}_\theta(x)$ estimate and the function to be regressed y. x is used to denote the independent variable. The cost is reduced by varying the parameters θ of $\mathcal{F}$.

Least squared error or L2 loss is a common choice for regression and penalises the residual quadratically. Since the cost function is quadratic, the gradient is steeper and converges faster to the optimal point for gradient-based approaches. Moreover, the loss function is smooth and the Hessian is guaranteed to be positive definite provided that there is sufficient data to estimate all parameters in the domain of the problem. If the L2 loss is used to penalise the magnitude of the parameters of the model, this is known as Tikhonov regularisation and can make a semi-definite matrix invertible. In the limit of the penalty factor going to zero, the inverse approaches the solution for the Moore-Penrose pseudo-inverse; however, the condition number of the matrix also tends towards infinity and increases the sensitivity of the solution to changes in the parameters. The benefit of this over the true pseudo-inverse is that it can be significantly cheaper to compute. But as with all forms of regularisation, the penalty factor of the regularisation can significantly change the optimal solution. The regularisation also avoids overfitting, as large parameter values are typically indicative of this. Linear least squares is applying the least squared error loss function to a linear-in-the-parameters function. The different types of linear least

squares differentiate the relationships between the residuals. If the residuals are identically weighted then it is referred to as Ordinary Least Squares (OLS).

Least Squared Error

$$f(\theta) = ||y - \mathcal{F}_\theta(x)||_2^2 \tag{15.5}$$

Least absolute error or L1 loss is also a common choice for many applications and is typically useful when the dataset contains many outliers. A common use for the L1 loss is for regularisation where it is often known as least absolute shrinkage and selection operator (LASSO). The benefit of L1 as a regulariser is that it enables model reduction as the L1 loss enables the parameters to be equal to zero, whereas the L2 loss only approaches it asymptotically. For the parameters to obtain a zero value, not all optimisers of the L1 loss are suitable (e.g. gradient descent). A proximal-point algorithm is an example of an appropriate algorithm. The L1 loss, in contrast to L2 loss, is not smooth and results in the Hessian being positive semi-definite as the gradient is linear.

Least Absolute Error

$$f(\theta) = ||y - \mathcal{F}_\theta(x)||_1 \tag{15.6}$$

Huber loss is a hybrid between L1 and L2 loss and is more robust to outliers than L2 and better fitting than L1 but also results in a semi-definite Hessian in some parts of its domain.

Huber Loss

$$f(\theta) = \begin{cases} ||y - \mathcal{F}_\theta(x)||_2^2, & |r| < 1 \\ ||y - \mathcal{F}_\theta(x)||_1, & \text{otherwise} \end{cases} \tag{15.7}$$

15.3. Separability

Separability is a determined by the problem being solved. For unconstrained problems, separability is determined by whether the cost function $f(\theta)$ can be decomposed into a sum of independent cost functions as given in Eq. 15.8. As a result of the cost functions being independent, each cost function can be optimised independently and in parallel. For constrained optimisation, depending on the method chosen, the constraints must be similarly decomposed. Block separability is the case where the function can be decomposed but into fewer than n blocks. In the case of multivariate B-splines, the cost functions are block separable but the constraints between the splines, albeit local, are not.

$$f(\theta) = \sum_{i=1}^{n} f_i(\theta_i) \tag{15.8}$$

15.4. Derivative-free Methods

Derivative-free approaches are useful when the derivative of the loss function is not available and preferable when evaluating the loss function is cheap. They are also useful if the finite-difference approximation of the gradient is slower to compute or the loss function is too noisy such that it prevents a reliable estimate of its gradient. Some derivative-free approaches are Nelder-Mead, line search, and golden section search. When gradient approaches are feasible, they are often better as the search can be directed using gradient information leading to faster convergence. Additionally, information about the optimality of the solution can be determined based on the local gradient. Moreover, when operating in high-dimensional space, the curse of dimensionality can make it infeasible to effectively explore the domain to obtain the optimum. Derivative-free methods are also beneficial for non-convex

optimisation as they can escape local minima and are better equipped to find global minima. However, it should be noted that not all derivative-free approaches can handle non-convex problems. For example, golden section search assumes the loss function is convex. As a result of the optimisation problem being an ordinary least squares problem with a linear function approximator, the problem is convex and the derivative information is available and the test case is high-dimensional, it is determined that derivative-free approaches are not useful for further study.

15.5. Gradient-based Methods

Gradient-based methods utilise first or higher-order derivative information to optimise a cost function. These methods are often faster than derivative-free approaches as they can use this information to direct the search. The disadvantage of these approaches is that derivative information is required which is not always available. Moreover, the optimisation algorithms have more strict requirements that need to be satisfied to guarantee convergence to an optimal point. For example, non-convex problems will lead to the algorithms converging to local optima and multiple runs with different initial points are required to get better results. Although derivative-free approaches are also susceptible to converging to local optima.

15.5.1. Matrix conditioning

Matrix conditioning is not a gradient-based approach in itself but rather a technique that is often applied to improve the convergence rates and to improve the numerical stability of the algorithms. Matrix conditioning is determined by the condition number and is determined by the ratio of the maximum to the minimum singular value of a matrix (Eq. 15.9)[26]. Singular values are large if the matrix is nearly singular which can be seen as dividing by zero. In the case of optimisation, the Hessian is singular when the local gradient is linear or nearly linear. For OLS, this case only occurs when a parameter of the function approximator has little to no influence on the loss function. Since the function approximator is a multivariate simplex or simplotope B-spline, this equates to a data coverage problem in the domain of that parameter. Squaring a matrix, squares the condition number and should be avoided when possible. This occurs when forming the normal equations (Eq. 15.10). The product $A^T A$ should therefore be avoided.

$$\kappa = \frac{\sigma_{\max}}{\sigma_{\min}} \tag{15.9}$$

$$A^T A\theta = A^T y \tag{15.10}$$

One possible approach to avoid explicitly forming the product $A^T A$ is by using the QR decomposition, which decomposes the matrix A into an orthogonal matrix Q and an upper-triangular matrix R. By using the property that the transpose of an orthogonal matrix is also its inverse and expanding the terms, the problem simplifies significantly. The matrix R is upper-triangular and that means the inverse can be computed by back-solving.

$$\begin{aligned} \theta &= -(A^T A)^{-1} A^T y \\ &= ((QR)^T (QR))^{-1} (QR)^T y \\ &= (R^T Q^T QR)^{-1} R^T Q^T y \\ &= R^{-1} (R^T)^{-1} R^T Q^T y \\ &= R^{-1} Q^T y \end{aligned} \tag{15.11}$$

15.5.2. Bisection Search

Bisection method is a simplistic line search approach that applies only to convex problems. The approach commences by specifying a search boundary $[a, b]$ in 1D. The gradient of the function is then evaluated at the midpoint θ_i of the search region. If the gradient is negative, it must lie in the region bounded by $[\theta_i, b]$ else it must lie in the region $[a, \theta_i]$. As a consequence of the boundary being specified in 1D, the function must be evaluated along a search direction which is not specified by the algorithm.

15.5.3. Steepest Descent

Steepest descent takes an optimal step in the steepest direction which is given by the local gradient of the cost function. For OLS, the optimal one-step update in the direction of the gradient is trivial to compute. However, if the data is not normalised, the cost function will be elliptical. The ramifications of an elliptical cost function is that the steepest descent direction does not typically intersect the major axis at the centre of the ellipse, which in turn means that the optimisation process will be very slow to converge to the optimum. This fact can be seen in Eq. 15.12 where κ is the condition number given by Eq. 15.9[57]. Normalising the cost function (thereby reducing the condition number to 1) can be done by determining the eigenvectors although only one eigenvector needs to be computed to get a 1-step optimal solution. Nevertheless, it is typically far more computationally expensive than other simpler methods.

$$||e_i||_A \leq \frac{\kappa - 1}{\kappa + 1}^i ||e_0||_A \tag{15.12}$$

Coordinate Descent

Coordinate descent is an optimisation approach that optimises an objective function one variable at a time, for example using line search, while keeping all other variables constant. The approach is very simple to implement but can be slow as it optimises parameters one at a time and if they are correlated the optimal descent direction is a linear combination of the variables. Approaches have been taken to parallelise the algorithm[41],[3]; however, these algorithms are complicated when constraints are not separable. Liu[41], for example, utilises clipping in a hypercube, which cannot be used when constraints occur in multiple blocks. Adaptive coordinate descent[42] uses principle component analysis (PCA) to decouple dimensions. It should be noted that PCA is destructive to the sparse structure of the problem and transformations will be required when applying the constraints. Moreover, coordinate descent is limited to smooth cost functions (see Figure 15.1 for a counterexample).

Coordinate Descent

$$\underset{\theta_j}{\text{minimise}} \quad f(\theta_i^{k+1}, \theta_j, \theta_k^k), \quad \forall i \in [1, j), \forall k \in (j, n], \forall j \in [1, n] \tag{15.13}$$

Block coordinate descent is a modification whereby a subset of θ is chosen to be minimised simultaneously. This trades off the simplicity of coordinate descent with the optimality of simultaneously optimising all variables at once. Blocks can also be randomly selected each iteration thereby avoiding the problem in Figure 15.1; however, this adds complexity and makes it more difficult to determine convergence criteria as no improvement in one iteration can possibly be improved by another iteration using a different subset of θ.

15.5.4. Conjugate Gradient Descent

The conjugate gradient method is an improvement on traditional gradient descent approaches, such as steepest descent, by making the search direction Q-orthogonal to all previous search directions. This is performed using the Gram-Schmidt process for orthogonalisation and using Q-orthogonal residuals to

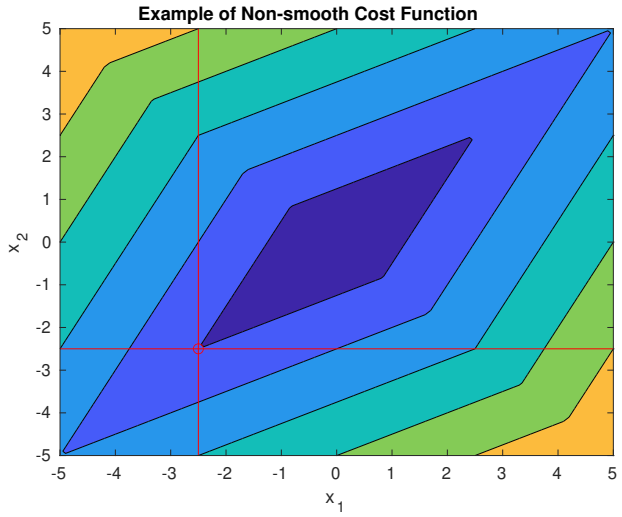


Figure 15.1: Coordinate descent will fail to reach the optimal cost when it reaches the red circle as performing line search in either direction will lead to a worse cost. The algorithm will then assume it has converged to the optimal value.

avoid needing to store all previous search directions[57]. Each new residual spans the Krylov subspace of one dimension higher, therefore after n searches, an orthogonal basis that spans $\mathbb{R}^n$ is created and the optimum is achieved. Of course, it is possible to stop before n iterations have occurred. However, it should be noted that the algorithm is used to determine the direction and not the step-size and must be combined with, for example, a line search or Newton's method. A practical issue arises when using Gram-Schmidt for orthogonalisation in finite-precision arithmetic, such as IEEE floating-point arithmetic, the Q-orthogonality is lost after several iterations and is significant for large n; however, this can be abated by resetting the search direction to the direction of steepest descent[55]. The real benefit of this approach is for sparse matrices. Finally, in the case of regression, the conjugate gradient method must be applied to the normal equations (Eq. 15.10) and since A^TA is formed, the condition number is squared.

15.5.5. Sequential Linear Programming

Sequential Linear Programming, also known as Successive Linear Programming, is an approach to nonlinear programming where the idea is to iteratively approximate the nonlinear problem as a linear problem, which is much simpler to solve. One issue with a linear approximation of a nonlinear function is that the approximation is only valid in the region of the linearisation. Trust regions are a modification that is commonly applied to ensure that the linear approximations are reasonably accurate and that the optimisation process is stable.

15.5.6. Newton's Method

Newton's method is a root finding algorithm that is also known as the Newton-Raphson method Eq. 15.14. The method works by using a linear Taylor series expansion of the function as the candidate for the fixed-point iteration. Graphically, the algorithm approximates the root by extrapolating the linear approximation of the gradient. In the case of optimising a function f, the optimal θ is when the gradient of the function is zero. Thus instead of finding the root of the function f, it can be applied to the derivative of f as in Eq. 15.15. The Newton method has some limitations, however, as it is attracted by both local minima and maxima and requires the existence of an invertible Hessian. The first problem can be solved by using an approximation that is close enough to a minima or an even more strict requirement that the Hessian is positive definite meaning there is only one optima and that the optima is a minima in all directions. Although there are also methods that deal with the saddle-point problem, see for example [12] and [50]. The latter problem has multiple solutions like the pseudo-inverse

and Tikhonov regularisation. Moreover, since the linear approximation of the gradient is applied to the first derivative and OLS is linear in the parameters, the approximation is exact. Therefore, the Newton update is a 1-step optimal update regardless of the initial estimate for the parameters. As a result, choosing $\theta_0 = 0$ means the computation can be simplified to Eq. 15.16. It should be noted that in the presence of noisy measurements the estimate of the Hessian, particularly when the data coverage is low, can cause large changes in the parameters as it will be modelling the noise. This can be abated by applying a threshold on the singular values when computing the pseudo-inverse.

$$\theta^{k+1} = \theta^k - \frac{f(\theta^k)}{f'(\theta^k)} \tag{15.14}$$

Newton's Method

$$\theta^{k+1} = \theta^k - \frac{f'(\theta^k)}{f''(\theta^k)} \tag{15.15}$$

$$\theta = -\frac{f'(\theta^0)}{f''(\theta^0)} \tag{15.16}$$

$$\theta = -(A^T A)^{-1} A^T y \tag{15.17}$$

15.5.7. Quasi-Newton Methods

Quasi-Newton methods operate under the same principle as Newton-methods; however, instead of computing the Hessian, approximations of the Hessian are utilised. The approximate forms can be computationally faster to compute or an approximation of the inverse can be computed to avoid matrix inversion in its entirety. These approaches also allow the Hessian to be updated more efficiently and may reduce the overall amount of memory required.

The secant method can be seen as applying Newton's method to the finite difference approximation of the derivative, or in this case to the second derivative to find the root of the derivative. This linear approximation of the Hessian is less accurate but the fact that $f'(\theta^{k-1})$ is already available can be exploited and reduce the complexity of computing the Hessian at the cost of slower convergence. In the case of OLS, the finite difference estimate would be exact as the Hessian is linear and the convergence should occur in one step. The finite difference from the previous time step can not be reused as the regression matrix A has changed. It should be possible to update the previous estimate with the new data without completely recomputing the previous derivative or the data can be reused with no modification as be used an inexact estimate.

Secant Method

$$\tilde{f}''(\theta^k, \theta^{k-1}) = \frac{f'(\theta^k) - f'(\theta^{k-1})}{\theta^k - \theta^{k-1}} \tag{15.18}$$

$$\theta^{k+1} = \theta^k - \frac{f'(\theta^k)}{\tilde{f}''(\theta^k, \theta^{k-1})} \tag{15.19}$$

Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shannon (BFGS) are variants of the secant method, which estimate the inverse of the Hessian and the Hessian, respectively[55]. The main purpose of computing the inverse of the Hessian is to determine good search directions and is identical to the conjugate-gradient method for quadratic problems[55]. These methods are particularly useful when functions of higher than second-order are being optimised and they are considered state-of-the-art.

Limited-memory BFGS (L-BFGS) is a modification to the BFGS algorithm that is aimed at large-scale optimisation, where storing all the matrices and data in memory is infeasible. The solution is to use rank-k approximations to the Hessian, where $k << n$.

15.6. Constrained Optimisation Approaches

This section will look into algorithms that can optimise the equality constrained quadratic programming problem that has been defined. Some methods are also able to handle inequality constraints with minimal modifications. Constrained optimisation algorithms are often meta-algorithms in the sense that they require unconstrained optimisation algorithms to run during some of the steps, which direct the unconstrained optimisation to a constrained optimal solution.

15.6.1. Kaczmarz Method

This method is a special case of Projection onto Convex Sets (POCS) or, alternatively, known as the method of alternating projections applied to affine constraints[59]. Kaczmarz is thus an iterative approach that can be applied to constraint satisfaction or regression. It is also known as the Algebraic Reconstruction Technique[59]. The method selects constraints either deterministically or stochastically. The most common deterministic approach is selecting the rows in a cyclic fashion. Stochastic approaches are typically better as they lead to faster convergence[59]. $\gamma \in (0, 1]$ is a relaxation parameter that controls the convergence speed and the sensitivity to noise. θ_0 is the unconstrained optimal solution.

$$\theta_i = \theta_{i-1} + \gamma \frac{y_i - A_i \theta_{i-1}}{||A_i||_2^2} A_i^T \tag{15.20}$$

The algorithm can also be seen as a special case of gradient descent[11], where i indicates the application of the i^{th} constraint. $y_i - A_i\theta$ is the residual and multiplying it by $-A_i^T$ produces a stochastic estimate of the gradient for OLS. The learning rate is predetermined and is set to γ and normalised by the L2 norm of A_i. This choice is based on the Lipschitz continuity of the first derivative which is bounded by the Hessian at the current iterate for a convex quadratic function. The predetermined learning rate is guaranteed to be stable and obviates the need to tune the hyper-parameter. Moreover, it is simple to compute. Much like gradient descent, convergence can be slow for correlated rows of A and a similar solution is to orthogonalise the rows to increase the convergence speed. Practical issues with orthogonalising the rows is treated more extensively in Section 15.5.4 about the Conjugate Gradient method but an important consideration is that orthogonalisation typically ruins sparsity thereby increasing computation time. As mentioned before, stochastically selecting the constraints avoids correlated updates and can speed convergence.

The downside to the Kaczmarz method is that the algorithm converges to an arbitrary point within the constraint set and not necessarily the orthogonal projection. This means the loss is no longer optimal. To solve this, Dijkstra's projection method should be used. A parallel formulation of Dijkstra's method was given in [25].

15.6.2. Null-space Projection

The equality constraint $H\theta = 0$ for the continuity equations implies that θ must lie in the null-space of H. Let $\Gamma = \text{null}(H)$, then θ is the projection of the unconstrained parameters $\tilde{\theta}$ into the null-space of H, as given by Eq. 15.21. By substituting Eq. 15.21 into the least squares problem, the equality constrained problem is transformed into the unconstrained problem in Eq. 15.22. The Hessian for Eq. 15.22 is given by Eq. 15.23. While A is block-diagonal sparse, $A\Gamma$ is not necessarily. This means the memory requirements are increased in order to store the dense matrix Γ but also that the Newton update that requires the inverse of the Hessian is no longer parallelisable and is dependent on the total number of parameters in the triangulation. This can be obviated by using the orthogonal projection of the Moore-Penrose pseudo-inverse, given by Eq. 15.24. The orthogonal projection obtained using this equation is no longer a basis for the projection but maintains the sparsity and is less susceptible to numerical precision errors. An alternative approach, used by De Visser[16] and Van den Aarsen[61], is to separate the constraint matrices into those that are internal and external to a partition, where a partition is a contiguous group of splines. De Visser applied the null-space projection to internal

constraints and used dual ascent for external constraint satisfaction. This has the benefit of reducing the amount of memory required which now scales with the size of the partitions. However, if inequality constraints are used, the process of determining the active set requires a recomputation of the null-space several times which is a very slow process.

$$\theta = \Gamma\tilde{\theta} \tag{15.21}$$

$$\underset{\tilde{\theta}}{\text{minimise}} \quad (y - A\Gamma\tilde{\theta})^T(y - A\Gamma\tilde{\theta}) \tag{15.22}$$

$$\frac{d^2 f}{d\theta^2} = \Gamma^T A^T A\Gamma \tag{15.23}$$

$$P(H) = I - H^+ H \tag{15.24}$$

15.6.3. Lagrange Multiplier Methods

The Lagrange multiplier methods introduce the concept of dual variables but are also referred to as Lagrange multipliers and are denoted by λ. The Lagrangian $\mathcal{L}$ of the problem of the form in Eq. 15.25 is given by Eq. 15.26.

$$\begin{aligned} \underset{\theta}{\text{minimise}} \quad & f(\theta) \\ \text{subject to} \quad & H\theta = c \end{aligned} \tag{15.25}$$

Lagrangian of a Function

$$\mathcal{L}(\theta, \lambda) = f(\theta) + \lambda^T(H\theta - c) \tag{15.26}$$

The dual function is given by Eq. 15.28 and the goal is to maximise $g(\lambda)$. The duality gap is given by Eq. 15.27. Strong duality means all constraints can be satisfied and means the duality gap should be zero. Simplex continuity constraints can always be satisfied; however, physical constraints may not always be and can lead to a positive duality gap (see proposition 1.1.4 and 1.1.5 in [3].) The duality gap is useful for determining a stopping criterion as it is indicative of the optimality of the solution.

$$\mathcal{D} = f(\theta) - g(\lambda) \tag{15.27}$$

$$g(\lambda) = \inf_{\theta} \mathcal{L}(\theta, \lambda) \tag{15.28}$$

Penalty Methods

Penalty methods is the most intuitive method method of Lagrangian methods as a large penalty means constraint violation is avoided. This simplicity is at the cost of numerical stability and can lead to the method satisfying constraints rather than optimising the cost function. A typical solution is to solve the unconstrained problem and iteratively increase the penalty factor until the process has converged and the constraints are satisfied. Interior and exterior-point methods are suited only for inequality constrained problems. The difference is whether or not the algorithm at any point of execution is within the feasible set. The aforementioned penalty approach is an exterior-point method as the initial point is not necessarily contained within the feasible set. Since interior-point methods require a feasible initial point, it is typically harder to initialise the algorithm.

Analytical Solution

An optimum of the Lagrangian implies the partial derivatives with respect to θ and λ should be equal to 0. Since θ is being minimised and λ is being maximised, the optimal point is a saddle point. These constraints are called the Lagrangian constraints if there are only equality constraints or, in the more general case, the Karush-Kuhn-Tucker (KKT) constraints.

$$\frac{\partial \mathcal{L}(\theta, \lambda)}{\partial \theta} = 0 \tag{15.29}$$

$$\frac{\partial \mathcal{L}(\theta, \lambda)}{\partial \lambda} = 0 \tag{15.30}$$

For ordinary least squares where $f(\theta)$ is given by Eq. 15.31, this leads to a closed-form solution Eq. 15.32, where the matrix being inverted is known as the KKT matrix. Note that if the constraint matrix H or the Hessian $A^T A$ is rank deficient, the KKT matrix will be singular and the Moore-Penrose pseudo-inverse should instead be used, which is often the case. It should be noted that the matrix inversion is of size $(n+m) \times (n+m)$ where n is the number of coefficients and m is the number of constraints. For higher-order continuity and finer meshes, this can quickly grow large.

$$f(\theta) = \frac{1}{2}||y - A\theta||_2^2 = \frac{1}{2}(y - A\theta)^T(y - A\theta) \tag{15.31}$$

Analytical Solution

$$\begin{bmatrix} \theta^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} A^T A & H^T \\ H & 0 \end{bmatrix}^{-1} \begin{bmatrix} A^T y \\ d \end{bmatrix} \tag{15.32}$$

Implicit Iterative Method

An iterative method, based on the analytical solution to the CQP above, is given by [1]. The Lagrange multipliers are not explicitly updated as the terms are implicitly included in the equations. An initial estimate for the multipliers is used to initialise the algorithm and will determine the number of steps required to converge. The algorithm is not divergent due to poor initialisation for convex quadratic programs and can be set to the zero vector for the first run[1]. It may, however, be beneficial to explicitly compute the Lagrange multipliers for initialisation at subsequent time steps as the multipliers will not deviate significantly from those of previous iterations. ϵ is a small number, typically around 10^{-6}. The matrix K can be factored to allow for efficient backsolving; however, matrix factorisation algorithms are in the computational complexity set $\mathcal{O}(n^3)$ where n is the size of the matrix. The complete factorisations are typically not necessary for backsolving and, for example in MATLAB, it is possible to request an 'economy-size' decomposition to save some computational time. Nevertheless, the algorithm typically converges before the benefits of the factorisation are apparent. Moreover, the factorisation needs to be recomputed at each time-step as the regression matrix changes due to new data; however, rank-1 updates can be applied[26]. The matrix K is sparse; however, it is not block diagonal meaning the matrix inversion is dependent on the total number of parameters. One benefit of this approach is that the Hessian need not be invertible as long as it is positive definite on the null-space of the constraint matrix[1].

Algorithm 2: Iterative Algorithm

$K = \left(A^T A + \frac{1}{\epsilon} H^T H\right)^{-1}$
$\theta^0 = K \cdot \left(A^T y + \frac{1}{\epsilon} H^T d - H^T \lambda^0\right)$
while *Not Converged* **do**
  $\theta^{k+1} = K(A^T A \theta^k + \frac{1}{\epsilon} H^T d)$
end

Dual Ascent

Dual ascent is an alternative when it is computationally inefficient to invert the KKT matrix. It works by optimising the parameters θ and then performing a gradient ascent step for the dual. This decomposes the large KKT matrix into two smaller optimisation problems. The gradient of the dual function $g(\lambda)$ is given by the residual of the equality constraint[8]. Inequality constraints are handled by applying max operator as in Eq. 15.35 after computing the gradient ascent step and can be applied similarly to all other dual-based methods.

Dual Ascent

$$\theta^{k+1} := \underset{\theta}{\operatorname{argmin}}\ \mathcal{L}(\theta, \lambda^k) \tag{15.33}$$

$$\lambda^{k+1} := \lambda^k + \alpha^k(H\theta^{k+1} - c) \tag{15.34}$$

$$\lambda_{\leq}^{k+1} = \max(0, \lambda_{\leq}^{k+1}) \tag{15.35}$$

Dual Decomposition

If $f(\theta)$ is separable into a sum of independent cost functions, as in section 15.3, then each subset θ_i can be optimised separately. This algorithm is referred to as dual decomposition.

Dual Decomposition

$$\theta_i^{k+1} := \underset{\theta_i}{\operatorname{argmin}}\ \mathcal{L}(\theta_i, \lambda^k) \tag{15.36}$$

$$\lambda^{k+1} := \lambda^k + \alpha^k(H\theta^{k+1} - c) \tag{15.37}$$

Method of Multipliers

The method of multipliers introduces the concept of the augmented Lagrangian $\mathcal{L}_\rho$ which includes a new penalty factor ρ, which must be greater than 0. The augmented Lagrangian does not change the optimal point of the Lagrangian $\mathcal{L}$, since the additional term is zero when the equality constraint is satisfied. It does, however, influence the convergence rate by determining whether the focus is on an optimal cost function or a constraint feasible solution[61]. The benefit of the additional term is that the cost function no longer needs to be smooth or differentiable and the cost function can be $+\infty$[8]. The cost of this benefit is that separability is lost, and thus parallelisability.

Augmented Lagrangian

$$\mathcal{L}_\rho(\theta, \lambda) = f(\theta) + \lambda^T(H\theta - c) + \frac{\rho}{2}||H\theta - c||_2^2 \tag{15.38}$$

Method of Multipliers

$$\theta^{k+1} := \underset{\theta}{\operatorname{argmin}}\ \mathcal{L}_\rho(\theta, \lambda^k) \tag{15.39}$$

$$\lambda^{k+1} := \lambda^k + \rho^k(H\theta^{k+1} - c) \tag{15.40}$$

Alternating Direction Method of Multipliers

Alternating Direction Method of Multipliers (ADMM) makes the method of multipliers separable again by introducing additional auxiliary parameters that need to be optimised and therewith a new set of

constraints. The primal step is also split into two primal steps, where one set of primal parameters are kept fixed. In the limit, the two sets of primal variables will converge to the same values. The primal step cannot be separated into more than two steps unless the primal parameters are separable[10], which means two steps is sufficient in any case. ADMM has been used for multivariate simplotope B-splines in [61] and [30]. Wei proposed a distributed[65] and an asynchronous[66] version of ADMM but this has yet to be applied to multivariate B-splines.

$$\begin{aligned} \underset{\theta}{\text{minimise}} \quad & f(\theta) + g(z) \\ \text{subject to} \quad & H\theta + Gz = c \end{aligned} \tag{15.41}$$

Alternating Direction Method of Multipliers

$$\theta^{k+1} := \underset{\theta}{\operatorname{argmin}} \ \mathcal{L}_\rho(\theta, z^k, \lambda^k) \tag{15.42}$$

$$z^{k+1} := \underset{z}{\operatorname{argmin}} \ \mathcal{L}_\rho(\theta^{k+1}, z, \lambda^k) \tag{15.43}$$

$$\lambda^{k+1} := \lambda^k + \rho^k(H\theta^{k+1} + Gz^{k+1} - c) \tag{15.44}$$

15.6.4. Mesh Smoothing

Mesh smoothing is not a typical approach to constrained least squares optimisation; however, the triangulation can be considered a mesh and smoothing filters, such as Laplacian smoothing or Taubin smoothing[60], can be applied to ensure smoothness between simplotopes in the triangulation. In contrast to typical meshes, triangulations contain duplicate points at the edges where two simplotopes meet. To solve this inconsistency, an average or weighted-average of the duplicate points can be used as a substitute for these points. It still needs to be determined how smoothing influences higher-order continuity for B-splines. Too many iterations of the smoothing filters can cause loss of relevant model nonlinearities. These methods can possibly be combined with other approaches by using these as fast smooth approximations for the initialisation of the iterative algorithms.

15.7. Online Optimisation Algorithms

Online algorithms are an important subset of optimisation algorithms, particularly in the field of system identification and adaptive control. They enable efficient updating of a previous solution with new data. A brief comparison of algorithms is provided in Table 15.1.

Table 15.1: A comparison of online algorithms.

Method	Memory Req.	Hyper-parameters	Sparsity	Noise Sensitivity	Rate of convergence	Computational Complexity
Recursive Least Squares	n^2	No	Yes (block diagonal)	Yes, depending on the rank of the Hessian, the amount of available data, and the spatial distribution of the data - which gets worse with an increased number of dimensions.	O(1)	High
Gradient Descent	n	Yes, the learning rate.	No, but vector addition and subtraction is trivially parallelisable and memory requirements are significantly lower.	No	O(1/k)	Low

15.7.1. Warm-starting

Warm-starting is an approach that is typically applied to iterative optimisation algorithms where an initial estimate is required. By utilising the optimal parameters from the previous iteration, it is assumed that the solution to the new problem with the new data is close to the old solution this in turn speeds up the convergence to the new optimal solution.

15.7.2. Online Gradient Descent

Online gradient descent is the application of gradient descent to the online setting. It estimates the gradient of the cost function using the new data as an estimate for the gradient of the cost function. However, the parameters are only updated in the direction of the gradient by a small amount given by the learning rate α. As time progresses, the parameters approach their expected values. As with all gradient descent approaches, it is typically difficult to select a good learning rate that results in smooth but fast learning. Techniques like momentum are useful for accelerating the learning while being stable. See [54] for an overview of the most popular modifications to gradient descent for adaptively selecting the learning rate; however, it has been observed that adaptive optimisation algorithms tend to overfit to the training data and do not generalise well[67]. In [37], they propose switching between adaptive methods and stochastic gradient descent (SGD) to gain the benefits of faster learning while avoiding overfitting. SGD and SGD with momentum also has the favourable property of converging to the minimum norm solution[67]. Finally, the memory requirements of gradient descent ($\mathbb{R}^n$) are lower than that for Hessian-based approaches ($\mathbb{R}^{n\times n}$), as only the parameter values need to be stored.

15.7.3. Recursive Least Squares

Recursive Least Squares (RLS) is a modification to Newton's method where the matrix inversion lemma is used to simplify the inversion of the Hessian when new data arrives. The first term in the inversion is actually a scalar and therefore the inversion simplifies to scalar division. The algorithm requires an initial estimate for the parameters θ_0 and the covariance P_0. a_t is the new row for the regression matrix A at time t. If no estimate is known a priori, it is typical to set the initial parameters to zero and the estimate of the covariance to a large positive number k multiplied by the identity matrix. It should be noted that if the positive number is too large, after a few updates with insufficient data coverage it can result in a badly conditioned matrix. To adapt to failure, the covariance matrix matrix can be reset or increased again by some large factor kI.

Algorithm 3: Recursive Least Squares

1 **Init:** $\theta_0 = \vec{0}$, $P_0 = kI$
2 $K_t = P_{t-1}a_t^T(a_tP_{t-1}a_t^T + I)^{-1}$
3 $\theta_t = \theta_{t-1} + K_t(y_t - a_t\theta_t)$
4 $P_t = P_{t-1} - K_ta_tP_{t-1}$

Exponentially-weighted recursive least squares is a modification to RLS that weights recent data higher than older data. The parameter that controls the weighting is μ and is also referred to as the forgetting factor. This approach is particularly useful for time-variant models or, in the case of aerodynamic system identification, for adapting to a change in the aerodynamic model after a structural or mechanical failure and it results in a smoother change of the model. Whether or not this is desirable is application dependent.

Algorithm 4: Exponentially-weighted Recursive Least Squares

1 **Init:** $\theta_0 = \vec{0}$, $P_0 = kI$
2 $K_t = P_{t-1}a_t^T(a_tP_{t-1}a_t^T\mathbf{+\mu})^{-1}$
3 $\theta_t = \theta_{t-1} + K_t(y_t - a_t\theta_t)$
4 $P_t = P_{t-1} - K_ta_tP_{t-1}$

A constrained approach to RLS only differs with the unconstrained case in the initial condition for the parameters and the covariance matrix[68]. However, Zhu[68] recommends a modification to the standard algorithm by applying the null-space projection matrix Γ just before updating the parameters to ensure the parameters still lie in the null-space of constraints, otherwise the numerical errors will increase with each subsequent iteration and the constraints will no longer be satisfied. Zhu also proposes a modification to handle inequality constraints by running the algorithm for all possible combinations of the active set which can be infeasible if the total number of inequality constraints is large (2^q initial conditions, where q is the number of inequality constraints).

Unconstrained Initialisation

$$P_0 = (A_0^T A_0)^{-1} \tag{15.45}$$
$$\theta_0 = P_0 A_0^T y \tag{15.46}$$

Constrained Initialisation

$$P_0 = (\Gamma^T A_0^T A_0 \Gamma)^+ \tag{15.47}$$
$$\theta_0 = H^+ d + P_0 A_0^T (y - A_0 H^+ d) \tag{15.48}$$

Algorithm 5: Robust Recursive Least Squares

1 **Init:** $\theta_0 = \vec{0}$, $P_0 = kI$
2 $K_t = P_{t-1} a_t^T (a_t P_{t-1} a_t^T + I)^{-1}$
3 $\theta_t = \theta_{t-1} + \mathbf{\Gamma[}K_t(y_t - a_t\theta_t)\mathbf{]}$
4 $P_t = P_{t-1} - K_t a_t P_{t-1}$

15.8. Multi-grid Methods

Multi-grid methods are a useful technique for iterative methods. They are based on solving the problem at varying scales as a good initialisation for finer grids. Moreover, they enable constraints to propagate faster through a mesh and this is one of their main uses in finite element methods. In the case of multivariate B-splines, the constraints may be physical constraints or continuity constraints between the simplices. Physical constraints should be satisfied but these are also typically inequality constraints in contrast to the continuity constraints which means that slack is introduced and that the constraints are not necessarily in the active set of constraints. If an inequality constraint is violated, multi-grid methods would aid in ensuring the constraint is satisfied in a globally optimal manner. However, the focus is placed on the continuity constraints and these will always benefit from multi-grid methods. It is, however, argued that the continuity constraints are an artifact of using splines and having a globally optimal solution to the problem subject to all the constraints is computationally expensive and weights data external to the current spline; equally to the more recent data, which may be more accurate if a failure has occurred or if the model is time-varying. In addition, the changes in the model are expected to small, except after a failure, and therefore convergence should be reasonably fast. In the case of a failure, the old data becomes invalid, which means the system would need to be re-excited in the domains of all the splines before the multi-grid approach is useful and not performing pure extrapolation across the entire domain of the triangulation. Finally, multi-grid methods are also affected by the curse of dimensionality, which increases the memory and computational requirements exponentially and therefore makes them infeasible for onboard online aerodynamic system identification.

16

Conclusions

Orthotope B-splines are better for high-dimensional regression as the algorithms required to implement them are simpler and generalise better than those for high-dimensional simplex splines. In this thesis, oblique projections and first-order optimisation with early stopping were demonstrated to improve results by avoiding overfitting - which is often the case in high-dimensional space - and avoiding constraints to be applied between splines when one spline is missing data which reduced the training and validation accuracy. Moreover, orthotope B-splines with a different memory organisation has the potential to significantly improve performance, on both a CPU and a GPU. Finally, a conflict-free use of shared memory on the GPU is critical to enable the scalability and parallelisability of the algorithm. Moreover, data coverage along with scalability are the two most important considerations for high-dimensional function approximation.

> What is the impact of the hyper-parameters on the optimiser's performance on the validation dataset and how does this compare to Newton's method?

The first research question was answered in chapter 5. It was determined that coarser grids are better for validation performance; however, it was also determined that the proposed algorithm is significantly less sensitive to grid choices than Newton's method. Consequently, the proposed algorithm is more robust than Newton's method. The continuity order has a lower impact on the proposed algorithm's validation performance but lower-order continuity is typically marginally better.

> Does the quality of the unconstrained input to the constrained optimiser affect the performance of the algorithm?

The second research questions was also answered in chapter 5. It was determined that overfitting the model when computing the unconstrained optimal solution leads to worse performance than Newton's method. In contrast, with early stopping and gradient descent, it was observed to significantly outperform Newton's method and with a lower cross-validation variance in the estimated model. With acceleration or momentum, such as ADAM or Nesterov accelerated gradient, the convergence rate of gradient descent can be significantly improved and makes it also a viable choice for online algorithms.

> Can oblique projections improve model accuracy on the validation dataset when compared to orthogonal projections when data coverage is low?

In chapter 6, it was determined that oblique projections are necessary to improve model accuracy on both the training and validation datasets. It was, however, determined that oblique projections

introduce a bias in the multi-dimensional case that changes the solution based on the order chosen for the projections.

> Can duplicated B-coefficients along boundaries be eliminated when applying constraints?

In chapter 12, it was determined that duplicated B-coefficients can be removed and can greatly improve scalability of the algorithm. However, it was also noted that deduplication does not work well with the bank conflict avoidance scheme selected as it does not generalise as well beyond first-order continuity due to the limit of 32 banks.

> Can the theoretical cache performance be improved by resorting the coefficients in the tessellation?

Depending on the order of continuity, it was shown in section 8.1 that the sorting of the B-coefficients can significantly improve computational speed as the L1 cache does not become as polluted and false sharing can be avoided when distributing the data to the processors. Additionally, fewer transactions are required between main memory and the caches and the reads can be coalesced. Moreover, the indexing functions simplify leading to a simpler and faster algorithm with fewer branches.

> Can orthogonal constraints be used to improve the speed of the algorithm at the cost of more memory usage?

In chapter 7, it was proven that 1D orthogonal constraints are beneficial and require less operations than required for two iterations of the non-orthogonal algorithm - which will typically not converge that fast - and is therefore always the better choice. Additionally, all the required model parameters are already read from memory for the lower-order continuity equations and therefore does not increase the memory load. Higher-dimensional orthogonality results in significant communication overhead between threads and prevents effective parallelisation of the algorithm and is, as a result, not feasible. Additionally, it was noted that once convergence has occurred in a dimension, projections in other dimensions do not lead to constraint divergence.

> Can bank conflicts in shared memory on the GPU be avoided to improve performance of the algorithm?

Bank conflicts were addressed in section 8.2. It was determined that bank conflicts are a critical barrier to effective parallelisation of the algorithm but was solved using a tiling of bank-conflict-free cubes to fill the hyper-dimensional planes. It requires certain directions within the cube to be handled separately but results in no bank conflicts in any dimension.

> "The aim of this research is to contribute to safer future forms of air transport, like delivery drones and air taxis, with the aid of robust adaptive flight control systems by investigating constrained optimisation algorithms for smooth high-dimensional and online system identification that can be executed in a parallel and asynchronous manner."

By answering the aforementioned research questions, this research has contributed to the body of knowledge for parallel constrained optimisation that can be used in the field system identification for robust adaptive flight control.

17

Recommendations

This section is to share potential further research opportunities in the context of the presented research.

- Directional derivatives and physical constraints can be used to improve model accuracy; however, overfitting and noise may be more of a concern. This research could also evaluate the impacts of these modifications in the online setting.
- Null-space projection using first-order optimisation may be a competitive alternative to first-order optimisation with dual ascent and can be used to initially optimise the Lagrange multipliers, if more flexibility in the optimality of the optimisation is required. Alternatively, an optimised ADMM approach could be explored.
- It was noted the first projection direction biases the constrained solution to favour the initial projection directions over the latter directions. Research into a weighted solution that minimises this bias should be evaluated to see if it influences model accuracy.
- Quality estimates for determining the weights when performing oblique projections has the potential to increase the flexibility and accuracy of the algorithm.
- A CUDA implementation can be useful to run larger models and enable online experiments to be performed to evaluate the algorithm in situ.

Bibliography

[1] Gerard Awanou, Ming-Jun Lai, and Paul Wenston. The multivariate spline method for scattered data fitting and numerical solutions of partial differential equations. In Guanrong Chen and Ming-Jung Lai, editors, *Wavelets and Splines: Athens 2005*, pages 24–75. Nashboro Press, 2005.

[2] Dimitri P. Bertsekas. *Nonlinear Programming*. Optimization and Computation Series. Athena Scientific, second edition, 1999. ISBN 1-886529-00-0.

[3] Dimitri P. Bertsekas. *Convex Optimization Algorithms*. Optimization and Computation Series. Athena Scientific, first edition, 2015. ISBN 978-1-886529-28-1.

[4] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Optimization and Computation Series. Athena Scientific, 1997. ISBN 1-886529-01-9.

[5] Hisham Bin Zubair. *Efficient Multigrid Methods based on Improved Coarse Grid Correction Techniques*. PhD thesis, Delft University of Technology, September 2009. URL `http://resolver.tudelft.nl/uuid:e0b43b38-ad07-4076-baf1-aa02579c397f`.

[6] Boeing. Statistical summary of commercial jet airplane accidents: Worldwide operations 1959-2018. Online, September 2019. 50th Edition.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009. ISBN 978-0-521-83378-3.

[8] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

[9] Anna E. Brunner. *Spline-based Wavefront Reconstruction*. PhD thesis, Delft University of Technology, December 2018.

[10] Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 155(1-2):57–79, January 2014. doi: 10.1007/s10107-014-0826-5.

[11] Xuemei Chen. Kaczmarz algorithm, row action methods, and statistical learning algorithms. *Contemporary Mathematics*, 706, 2018.

[12] Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganuli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, 2014. URL `http://papers.nips.cc/paper/5486-identifying-and-attacking-the-saddle-point-problem-in-high-dimensional-non-convex-optimization`.

[13] C.C. de Visser, Q.P. Chu, and J.A. Mulder. Differential constraints for bounded recursive identification with multivariate splines. *Automatica*, 47(9):2059–2066, September 2011. doi: 10.1016/j.automatica.2011.06.011.

[14] Coen C. de Visser. *Global Nonlinear Model Identification with Multivariate Splines*. PhD thesis, Delft University of Technology, 2011. URL `http://resolver.tudelft.nl/uuid:6bc0134a-0715-4829-903d-6479c5735913`.

[15] Cornelis C. de Visser and Michel Verhaegen. Wavefront reconstruction in adaptive optics systems using nonlinear multivariate splines. *Journal of the Optical Society of America A*, 30(1):82–95, January 2013. doi: 10.1364/josaa.30.000082.

[16] Cornelis C. de Visser, Elisabeth Brunner, and Michel Verhaegen. On distributed wavefront reconstruction for large-scale adaptive optics systems. *Journal of the Optical Society of America A*, 33 (5):817–831, May 2016. doi: 10.1364/JOSAA.33.000817.

[17] Kenneth M. Dorsett and David R. Mehl. Innovative control effectors. techreport, Lockheed Martin Tactical Aircraft Systems, January 1996. URL `https://apps.dtic.mil/dtic/tr/fulltext/u2/b212813.pdf`.

[18] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[19] Richard L. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, December 1983. doi: 10.2307/2288193.

[20] Richard L. Dykstra and James P. Boyle. An algorithm for least squares projections onto the intersection of translated, convex cones. *Journal of Statistical Planning and Inference*, 15:391–399, 1986-1987. doi: 10.1016/0378-3758(86)90111-4.

[21] European Aviation Safety Agency. Executive director decision 2015/012/r. Online, May 2015.

[22] European Aviation Safety Agency. Explanatory note to decision 2015/012/r. Online, May 2015.

[23] Federal Aviation Administration. Qualification, service, and use of crewmembers and aircraft dispatchers. Online, November 2013. URL `https://www.govinfo.gov/content/pkg/FR-2013-11-12/pdf/2013-26845.pdf`.

[24] Guilherme França, Daniel P. Robinson, and René Vidal. Admm and accelerated admm as continuous dynamical systems. 2018.

[25] Norbert Gaffke and Rudolf Mathar. A cyclic projection algorithm via duality. *Metrika*, 36:29–54, 1989. doi: 10.1007/BF02614077.

[26] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, fourth edition, 2013. ISBN 978-1-4214-0859-0.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[28] Richard J. Hanson and Charles L. Lawson. Extensions and applications of the householder algorithm for solving linear least squares problems. *Mathematics of Computation*, 23(108):787–812, October 1969. ISSN 1088-6842. doi: 10.2307/2004965.

[29] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, second edition, 2009. ISBN 978-0-387-84858-7. doi: 10.1007/978-0-387-84858-7.

[30] Bart Hooij. Online distributed approach for aerodynamic model identification of the ice aircraft. Master's thesis, Delft University of Technology, 2020. URL `http://resolver.tudelft.nl/uuid:a92817a5-b7f9-4d5f-84de-30f6bd78f668`.

[31] Florian J.A. Huisman. Full flight envelope aerodynamic modelling of the cessna citation ii using physical splines. Master's thesis, Delft University of Technology, 2017. URL `http://resolver.tudelft.nl/uuid:f9926c24-e86b-4eee-8986-d427fb4bc667`.

[32] International Air Transport Association. *Guidance Material and Best Practices for the Implementation of Upset Prevention and Recovery Training*. International Air Transport Association, second edition, 2018. ISBN 978-92-9229-841-8.

[33] International Air Transport Association. *Loss of Control In-Flight Accident Analysis Report*. International Air Transport Association, 2019. ISBN 978-92-9264-002-6.

[34] Stephen A. Jacklin. Closing the certification gaps in adaptive flight control software. In *AIAA Guidance, Navigation and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, August 2008. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20090026333.pdf.

[35] S. Kaczmarz. Angenäeherte auflösung von systemen linearer gleichungen. *Bull. Int. Acad. Polon. Sci. A.*, pages 355–357, 1937.

[36] R. Karagöz. Tensor network b-splines for high-dimensional function approximation. Master's thesis, Delft University of Technology, April 2020. URL http://resolver.tudelft.nl/uuid:d637a2f4-512d-4330-83ae-29f7ef2958ed.

[37] Nitish S. Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. Technical report, Salesforce Research, December 2017. URL https://arxiv.org/abs/1712.07628.pdf.

[38] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, May 2015. URL https://arxiv.org/abs/1412.6980.

[39] Vladislav Klein and Eugene A. Morelli. *Aircraft System Identification Theory and Practice*. AIAA Education. American Institute of Aeronautics and Astronautics, 2006. ISBN 1-56347-832-3.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[41] Ji Liu, Stephen J. Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *Journal of Machine Learning Research*, 16:285–322, 2015. ISSN 1533-7928. URL http://www.jmlr.org/papers/v16/liu15a.html.

[42] Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Adaptive coordinate descent. In *GECCO '11: Genetic and Evolutionary Computation Conference*, number 13, pages 885–892. Association for Computing Machinery, July 2011. doi: 10.1145/2001576.2001697.

[43] Jonathan Lévy, Shanshan Ren, Hamid Mushtaq, Koen Bertels, and Zaid Al-Ars. GASAL2: a GPU accelerated sequence alignment library for high-throughput NGS data. *BMC Bioinformatics*, 2019. doi: https://doi.org/10.1186/s12859-019-3086-9.

[44] I. Matamoros and C.C. de Visser. Incremental nonlinear control allocation for a tailless aircraft with innovative control effectors. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, January 2018. doi: 10.2514/6.2018-1116.

[45] Ion Matei and John S. Baras. Nonlinear programming methods for distributed optimization. Technical Report v1, July 2017. URL https://arxiv.org/abs/1707.04598v1.

[46] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989. doi: 10.1016/S0079-7421(08)60536-8.

[47] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Adrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236.

[48] Douglas W. Moore. *Simplicial Mesh Generation with Applications*. PhD thesis, Cornell University, August 1992.

[49] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence O(1/k2). *Doklady AN USSR*, 269:543–547, 1983.

[50] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, second edition, 2006.

[51] NVIDIA. Nvidia turing gpu architecture. Technical report, NVIDIA, 2018. URL https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf.

[52] NVIDIA. Cuda c++ best practices. Online, 2020. URL https://docs.nvidia.com/cuda/pdf/CUDA_C_Best_Practices_Guide.pdf.

[53] NVIDIA. Turing tuning guide. Online, 2020. URL https://docs.nvidia.com/cuda/pdf/Turing_Tuning_Guide.pdf.

[54] Sebastian Ruder. An overview of gradient descent optimization algorithms. Technical Report v2, Insight Centre for Data Analytics, 2016. URL https://arxiv.org/abs/1609.04747v2.

[55] Andrzej Ruszczyński. *Nonlinear Optimization*. Princeton University Press, first edition, 2006. ISBN 978-0-691-11915-1.

[56] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003. ISBN 978-0-89871-800-3.

[57] Jonathan R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical Report 1.25, Carnegie Mellon University, 1994.

[58] Georgios Smaragdos, Georgios Chatzikonstantis, Rahul Kukreja, Harry Sidiropoulos, Dimitrios Rodopoulos, Ioannis Sourdis, Zaid Al-Ars, Christoforos Kachris, Dimitrios Soudris, and Chris I. De Zeeuw. Brainframe: a node-level heterogeneous accelerator platform for neuron simulations. *Journal of Neural Engineering*, 2017. doi: https://doi.org/10.1088/1741-2552/aa7fc5.

[59] Thomas Strohmer and Roman Vershynin. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15:262–278, 2009. doi: 10.1007/s00041-008-9030-4.

[60] Gabriel Taubin. Curve and surface smoothing without shrinkage. In *Proceedings of IEEE International Conference on Computer Vision*, pages 852–857. IEEE, June 1995. doi: 10.1109/ICCV.1995.466848.

[61] M.S.T. van den Aarssen. Distributed approach for aerodynamic model identification of the ice aircraft. Master's thesis, Delft University of Technology, 2018. URL http://resolver.tudelft.nl/uuid:df32613c-5f3c-484e-84ea-99672939d6a5.

[62] M.A. van den Hoek, C.C. de Visser, and D.M. Pool. Identification of a cessna citation ii model based on flight test data. In Boguslaw Dolega, Robert Glebocki, Damian Kordos, and Marcin Zugaj, editors, *Advances in Aerospace Guidance, Navigation and Control*, volume 4, pages 259–278. CEAS, Springer, April 2017. doi: 10.1007/978-3-319-65283-2. URL http://resolver.tudelft.nl/uuid:c0a57513-38b7-4d3a-898c-fa57c3e7ac2e.

[63] I.V. van der Peijl. Physical splines for aerodynamic modelling of innovative control effectors. Master's thesis, Delft University of Technology, April 2017. URL http://resolver.tudelft.nl/uuid:7730dd69-2bb8-4047-8f6d-5681cfd04cee.

[64] T. Visser, C.C. de Visser, and E. van Kampen. Quadrotor system identification using the multivariate multiplex b-spline. In *AIAA Atmospheric Flight Mechanics Conference*. AIAA, January 2015. doi: doi.org/10.2514/6.2015-0747. URL http://resolver.tudelft.nl/uuid:6e50cda5-3c14-415e-b800-3e1dd42102b5.

[65] Ermin Wei and Asuman Ozdaglar. Distributed alternating direction method of multipliers. In *2012 IEEE 51st IEEE Conference on Decision and Control*, pages 5445–5450. IEEE, December 2012. doi: 10.1109/CDC.2012.6425904.

[66] Ermin Wei and Asuman Ozdaglar. On the O(1/k) convergence of asynchronous distributed alternating direction method of multipliers. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 551–554. IEEE, December 2013. doi: 10.1109/GlobalSIP.2013.6736937.

[67] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, 2017. URL http://papers.nips.cc/paper/7003-the-marginal-value-of-adaptive-gradient-methods-in-machine-learning.

[68] Y. Zhu and X. Rong Li. Recursive least squares with linear constraints. *Communications in Information and Systems*, 7(3):287–312, 2007. URL https://projecteuclid.org/euclid.cis/1201531976.