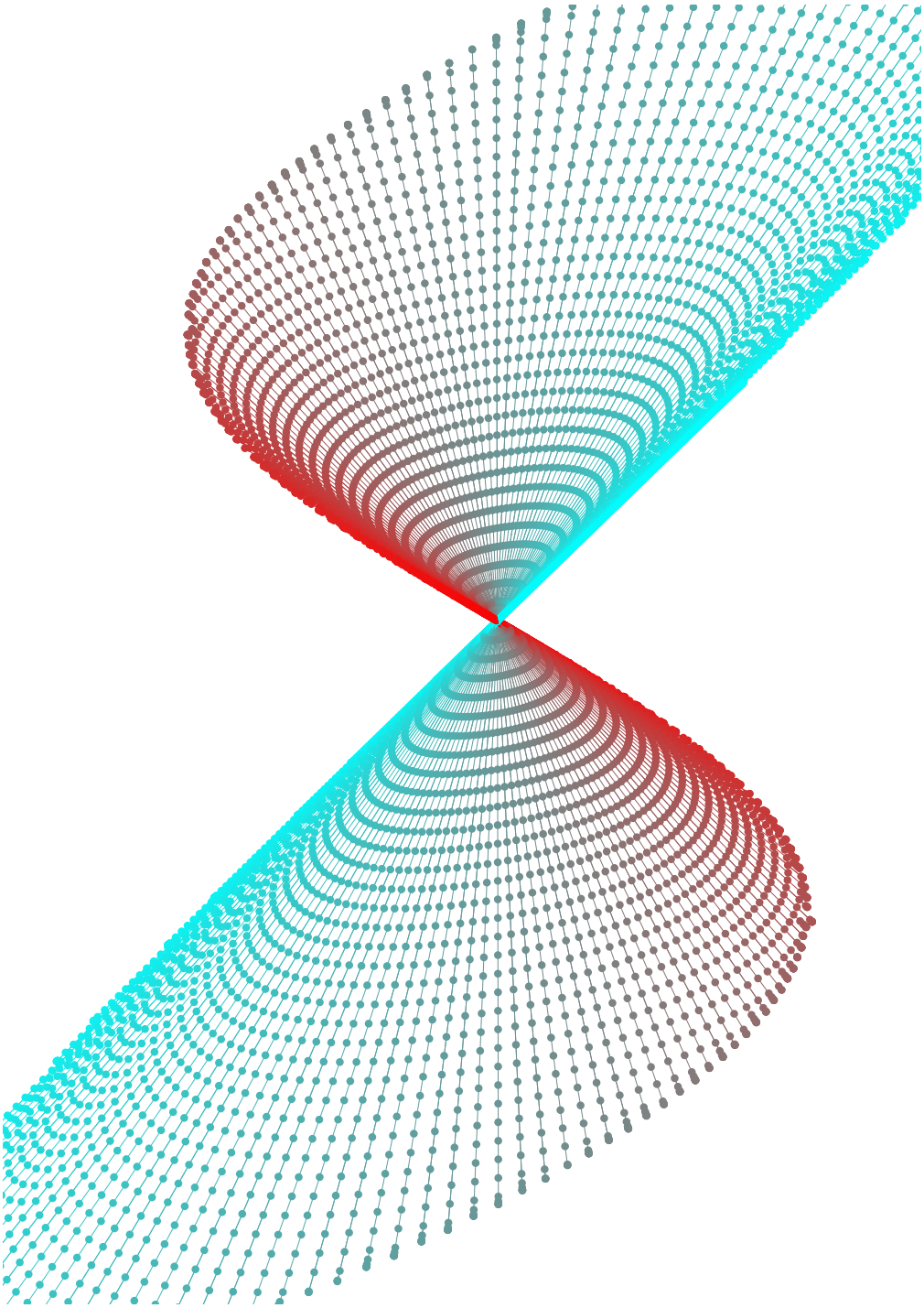
QUANTIFYING THE RANGES OF FEASIBLE CONTROL STRATEGIES FOR RESERVOIR LIFECYCLE OPTIMIZATION

Jayanth Krishnan Natarajan



TU Delft Delft University of Technology

Quantifying the ranges of feasible control strategies for reservoir lifecycle optimization

by

Jayanth Krishnan Natarajan

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday March 23, 2018 at 13:00.

Student number:	4506278	
Project duration:	December, 2016 – March, 2018	
Thesis committee:	Dr. Ir. Olwijn Leeuwenburgh,	TNO/TU Delft
	Prof. Dr. Ir. Jan Dirk Jansen,	TU Delft
	Dr. Hadi Hajibeygi,	TU Delft
	Prof. Dr. Ir. Arnold W. Heemink,	TU Delft

TU Delft

Abstract

Model based optimization of reservoir water flooding is an ill-posed problem where significantly different control strategies deliver near identical Net Present Values (NPVs). Discovering and exploiting the existence of "redundant" control strategies - particularly those in close proximity to an optimal strategy - is valuable since this offers operational flexibility in reservoir management. To identify such 'flexible strategies' this thesis proposes a workflow to characterize the space of feasible solutions. The feasible region or feasible solution space consists of all control strategies that deliver an NPV within some threshold from an optimal value.

Ensemble-based optimization is performed with strong Wolfe line search to identify an optimal control strategy. The BFGS scheme is used to iteratively approximate the Hessian matrix. One dimensional exploration is performed along the singular vector directions that characterize the null space of the Hessian. A thorough exploration results in an accurate characterization of the feasible region. Such an approach however is computationally intractable in case of realistic reservoirs with multiple hundred controls. To address this, a high-dimensional polytope is first defined using the end points from the exploration step. Subsequently, an innovative cross-section constrained maximum volume ellipsoid is inscribed within this polytope to generate an ellipsoidal approximation of the feasible region. Validation results are then presented which show that even one ellipsoid centred at the optimum control vector coordinate provides a conservative description of the feasible solution space.

Acknowledgements

To Ammi and Appu, (my parents) I could not have imagined completing this thesis without your moral and emotional support. Our daily Skype/"Ez"Talks calls were a constant source of motivation for me and I am immensely grateful to you.

I am also immeasurably grateful to Dr. Olwijn Leeuwenburgh, first for giving me this opportunity to work with him, second for patiently listening to me and all my ideas no matter how far-fetched and third for trying his best to keep me focused on one task at a time. I am pretty sure the last part was really difficult! I would also like to thank Prof. Jan Dirk Jansen for our initial meetings around mid-2016 which made me certain that I wanted to pursue a thesis in optimization and also for our many progress meetings later on.

This thesis has been a long and challenging journey for me. I am thankful to my friends in Delft, Den Haag and abroad who have supported me at every step.

Jayanth Krishnan Natarajan
March 22, 2018

Notation

a or A	a scalar (integer or real)
$a(.)$ or $A(.)$	a scalar function
$\boldsymbol{a}$	a vector
$\boldsymbol{a}(.)$	a vector function
a_i	element i of vector $\boldsymbol{a}$
$\boldsymbol{A}$	a matrix
$A_{i,j}$	element i, j of matrix $\boldsymbol{A}$
a	a random scalar variable
$\mathbf{a}$	a random vector
$\mathbf{A}$	a random matrix
$\nabla_{\boldsymbol{b}}\, \boldsymbol{a}$	gradient of $\boldsymbol{a}$ with respect to $\boldsymbol{b}$

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This report considers the model based optimization of reservoir water flooding. Waterflooding is the most common secondary recovery strategy for oil reservoirs and aims to induce flow of oil towards producing wells by providing pressure support through injection of water at injection wells. Since the placement of wells and the heterogeneity of the reservoir rock will make it difficult to say a priori which injection and production strategy would be most successful, simulation models can be used to investigate alternative scenarios. While this would undoubtedly provide valuable insight, it may not deliver a strategy that is optimal. Optimality in hydrocarbon recovery is commonly defined by the economics of the project, and quantified usually in terms of the Net Present Value (NPV). Numerical optimization methods can then be employed to search for the strategy that delivers the maximum NPV.

Injection and production strategies can be parametrized by for example well rates or well pressures at specified times or within specified time intervals. An alternative is to prescribe fixed well pressures (well head or bottom hole) and manipulate the well rate by the opening and closing of a valve. In this study the controls that parameterize the operating strategy are chosen to be valve settings. An optimization method is then required to determine those valve settings that result in maximum NPV. A large number of methods are available for this purpose, but experience has shown that gradient-based methods tend to be very efficient for reservoir optimization problems that typically consist of a large number of controls. A problem with this approach is that exact gradients require the use of an adjoint method that exploits the availability of derivatives that generally are not accessible for commercial simulators. The EnOpt (Ensemble Optimization) gradient initially introduced in Chen (2008) [5], Chen (2009) [6] and later improved upon as summarized by Fonseca (2017) [15] may therefore be a viable alternative. Although more inefficient in comparison to the adjoint gradient presented in Brouwer & Jansen (2004) [3], at least for deterministic optimization problems, the EnOpt gradient does not require simulator source code access. This is a major advantage in the context of commercial simulator usability.

Jansen et al. (2009) [25] observed that waterflood optimization is ill-posed as there exist more control variables than necessary. Mathematically an ill-posed problem is equivalent to a system of linear equations with infinitely many solutions. It has often been observed that many different control strategies deliver approximately identical NPVs. Significant value could exist in having multiple optimal solutions available. Firstly, it offers the possibility to select a solution that also optimizes a second objective. Van Essen et al. (2011) [40] proposed a methodology for hierarchical multi-objective optimization that exploits the existence of such multiple solutions. In later work Van Essen et al. (2012) [41] addressed the issue of flexibility. They noted that since reservoir simulation models are coarse and may not therefore sufficiently resolve the near-well dynamics that affect the observed behavior in the well, strategies suggested by such models may never be implemented in the field. They proposed a two-level closed-loop approach in which the reservoir model provides set points for a fast-model optimization step, with the idea that the fast model can be continuously adapted to real-time observed dynamics and therefore provide better operator guidance in the short term. The availability of flexibility is also important because of unforeseen circumstances that may arise during production. An example could be problems with wells that need to be taken offline for extensive maintenance. It may not be possible to update models and re-optimize strategies every time conditions in the field change.

An alternative approach to provide flexibility is to exploit the existence of multiple optimal operating strategies to identify those solutions that are exchangeable at some stage of the life cycle. A first step towards establishing a workflow that enables identification of 'flexible strategies' is to characterize the space of feasible solutions. We will define these solutions here as those control strategies that deliver an NPV value that is within some threshold from the optimal value.

Exploration of an optimal solution space has been investigated in different domains. The term 'design space exploration' was encountered frequently in microprocessor design (Gries (2003) [21], Eyerman et al. (2006) [10], Kang et al. [26] amongst many others) and also in aircraft design (Baker et al. [1]). However in these works, the focus is on identifying an optimal solution itself rather than characterization of the optimal solution space.

The work in systems biology by Zamora-Sillero et al. (2011) [44] aligns most closely with the goals of this thesis. In this work, a cost function is defined which calculates how well a set of biochemical parameters produces a certain behavior. The feasible space (which is recognized as potentially non-convex and poorly connected) is defined as that subset of biochemical parameter space where the cost associated with a parameter vector is below some threshold. To characterize the feasible space, global exploration termed OEAMC (out-of-equilibrium adaptive Metropolis Monte Carlo) sampling is performed first. The feasible samples found here serve as seed points for local exploration with MEBS (multiple ellipsoid based sampling). The MEBS approach iteratively constructs ellipsoids within which uniform sampling is performed. The goal with MEBS is to have multiple ellipsoids enclose nearly convex feasible sub-regions. The feasible samples inside each ellipsoid are then used to collectively characterize the feasible space. Uniform sampling within multiple ellipsoids therefore results in a 'fine-grained' characterization of the feasible space. Although ideal, such an approach is potentially intractable in the context of reservoir problems, which typically consist of multiple hundred dimensions.

We investigate here a variation on the above approach that addresses this shortcoming. In particular we implement a BFGS scheme to estimate the Hessian matrix containing second derivatives of the objective function with respect to the controls. Following Van Essen et al. (2011) [40] we define its null space and explore the principal directions of this space in a systematic way. Secondly, we avoid the computational cost that would be imposed by repeated uniform sampling in multiple ellipsoids by first defining a high-dimensional polytope and subsequently determining a cross-section constrained maximum volume inscribed ellipsoid. We then show that this (single) ellipsoid provides a reasonably conservative, if not thorough description of the feasible solution space.

The remainder of this report is structured as follows: In Chapter 2 we define the mathematical optimization problem as well as the objective function for our reservoir problem. We also review the fundamentals of optimization theory that will be required in later chapters as well as the ensemble optimization methodology. Chapter 3 discusses the steps used for exploration of the Hessian null space. Afterwards in Chapter 4 the ellipsoid fitting workflow is described in detail. Chapter 5 contains a description of experimental results that were obtained with a simple reservoir simulation model to which the entire workflow was applied. Finally, Conclusions and Recommendation for further work are provided in Chapter 6.

Chapter 2

Optimization

In this chapter the economic lifecycle optimization of a reservoir model is considered. Water flooding is selected as the method of oil recovery since it is a well understood process and can be simulated/modelled accurately over long distances and time spans. Water flooding is accomplished with injector wells which help maintain reservoir pressure and displace oil towards the producer. Each injector well is equipped with one ICV and the valve aperture can be controlled over time. Controllable wells are fundamental to the continuous (reservoir) management paradigm of closed loop reservoir management (CLRM). (Brouwer & Jansen (2004) [3]) Such wells enable improved oil recovery and profitability with decreased water production and delayed water breakthrough.

2.1 NPV Calculation

An objective function J is defined that tracks the discounted net present value (NPV) associated with the injected and produced liquid volumes at each time step.

$$J = \sum_{k=1}^{K} \left(\frac{\left\{ \left[Q_{O_k} \cdot r_O - Q_{W_{p_k}} \cdot r_{W_p} \right] - \left[Q_{W_{i_k}} \cdot r_{W_i} \right] \right\} \cdot \Delta t_k}{(1+b)^{\frac{t_k}{\tau_t}}} \right) \tag{2.1}$$

Q_{O_k}	Oil production rate	(m^3/day)
$Q_{W_{p_k}}$	Water production rate	(m^3/day)
$Q_{W_{i_k}}$	Water injection rate	(m^3/day)
r_O	Price of oil	$(\$/m^3)$
r_{W_p}	Cost of water produced	$(\$/m^3)$
r_{W_i}	Cost of water injected	$(\$/m^3)$
Δt_k	Difference between consecutive time steps	(days)
t_k	Cumulative time corresponding to time step k	(days)
τ_t	Reference time for discounting	(days)
b	Discount factor	(%/year)
k	Time step	[-]
K	Total Duration	(days)

Table 2.1: NPV calculation parameters

2.2 Optimization Formulation

The goal of optimization is to determine the ICV aperture settings that enable the highest NPV over the producing life of the reservoir. This maximization problem can be formulated as:

$$\begin{aligned} \underset{\boldsymbol{u}}{\textbf{Max}} \quad & J(\boldsymbol{u}), \\ \textbf{Subject to} \quad & \boldsymbol{u}^{min} \leqslant \boldsymbol{u} \leqslant \boldsymbol{u}^{max}, \end{aligned} \tag{2.2}$$

where $\boldsymbol{u}$ is a control vector consisting of the injector ICV aperture settings over time. Vectors $\boldsymbol{u}^{min}$ and $\boldsymbol{u}^{max}$ are simple bound constraints which specify the minimum and maximum valve apertures at each well for each time step. Consequently, the length of the control vector can be defined as:

$$\text{length}(\boldsymbol{u}) = (\text{ICV's per well}) \times (\#\text{ of wells}) \times (\#\text{ of control time steps}). \tag{2.3}$$

As an alternative to ICV apertures, flowing bottom hole pressures or well rates can also be specified as controls.

2.3 The EnOpt (Ensemble Optimization) gradient

This thesis uses gradient based optimization. The gradient $\boldsymbol{g}$ at coordinate $\boldsymbol{u}$ of the multi-variate (objective) function J represents the direction of fastest function rise and can be combined with the steepest ascent algorithm to reach a local optimum (maximum). (Nocedal & Wright (2006) [36] Section 2.2, p. 21)

The adjoint method is one approach to calculate the gradient for a reservoir model. This method is efficient in that it requires one forward simulation and just one additional backward (adjoint) simulation to determine the exact gradient, independent of the number of controls. (Brouwer & Jansen (2004) [3]) While efficient, this method also requires source code access for implementation - a task that is not feasible with commercial simulators - and is thus not considered.

Consequently, the approximate ensemble based gradient (EnOpt gradient) introduced by Chen (2008) [5] is used. With this, the gradient is calculated as a statistical (least squares) relation between an ensemble of perturbed control vectors and the associated objective function values. The size of the perturbed control ensemble decides the number of simulations required and also influences the gradient quality. (Fonseca et al. (2015a) [13])

This thesis uses a deterministic reservoir model for optimization where it is assumed the subsurface geology is already known (single geological realization). Uncertainty in subsurface geology relating to the distribution of fluid flow related parameters such as permeability and channel orientations can also be accounted for. This is done by creating an ensemble of equi-probable geological realizations in an approach known as robust optimization. (Fonseca et al. (2017) [15], Chen et al. (2009) [6] or Van Essen et al. (2009) [39])

2.3.1 Mathematics Of EnOpt

The mathematics of the EnOpt gradient is presented here based on the work in Fonseca et al. (2017) [15]

Here, $\boldsymbol{u}$ is a control vector consisting of d elements where each element is a variable to be optimized. These elements can either be ICV aperture settings, bottom hole pressures or rates. A user specified initial guess of $\boldsymbol{u}$ is supplied at the start of optimization.

$$\boldsymbol{u} = \begin{bmatrix} u_1, & u_2, & \ldots, & u_d \end{bmatrix}^T \tag{2.4}$$

At iteration l, a multivariate Gaussian distributed ensemble of size N_p is generated as $[\mathbf{u}_{l,1}, \mathbf{u}_{l,2}, \ldots, \mathbf{u}_{l,N_p}]$ with distribution mean as $\boldsymbol{u}_l$ and a predefined distribution-covariance matrix $\boldsymbol{C}_\mathbf{U}$. The control vector $\boldsymbol{u_l}$ is iteratively updated till convergence while the matrix $\boldsymbol{C}_\mathbf{U}$ is kept a constant.

The function value for each member of the perturbed ensemble is calculated as $J(\boldsymbol{m}, \mathbf{u}_{l,j})$ where $\boldsymbol{m}$ represents the geological realization considered and $j = 1, \cdots N_p$. Assuming the second derivatives of J are bounded and continuous, a truncated Taylor series expansion can be used to express the change in function value arising due to the perturbations in terms of derivatives as:

$$J(\boldsymbol{m}, \mathbf{u}_{l,j}) \approx J(\boldsymbol{m}, \boldsymbol{u}_l) + (\mathbf{u}_{l,j} - \boldsymbol{u}_l)^T \nabla J(\boldsymbol{m}, \boldsymbol{u}_l) + \mathcal{O}(\|\Delta \boldsymbol{u}\|^2), \tag{2.5}$$

where $\nabla J(\boldsymbol{m}, \boldsymbol{u}_l)$ is the $d \times 1$ gradient vector. Ignoring the error term $\mathcal{O}(\|\Delta \boldsymbol{u}\|^2)$, the first order approximation in equation 2.5 can be rewritten as:

$$(\mathbf{u}_{l,j} - \boldsymbol{u}_l)^T \nabla J(\boldsymbol{m}, \boldsymbol{u}_l) = J(\boldsymbol{m}, \mathbf{u}_{l,j}) - J(\boldsymbol{m}, \boldsymbol{u}_l). \tag{2.6}$$

Equation 2.6 suggests the difference in control vector coordinates and objective function values - relative to the optimum - introduced by the perturbation can be organized into a linear system of equations.

This is done using a $d \times N_p$ matrix $\Delta \mathbf{U}_l$ and an $N_p \times 1$ vector $\Delta \boldsymbol{j}_l$ where:

$$\Delta \mathbf{U}_l = \begin{bmatrix} \mathbf{u}_{l,1} - \boldsymbol{u}_l, & \mathbf{u}_{l,2} - \boldsymbol{u}_l & \ldots & \mathbf{u}_{l,N_p} - \boldsymbol{u}_l \end{bmatrix}, \tag{2.7}$$

$$\Delta \boldsymbol{j}_l = \begin{bmatrix} J(\boldsymbol{m}, \mathbf{u}_{l,1}) - J(\boldsymbol{m}, \boldsymbol{u}_l), & J(\boldsymbol{m}, \mathbf{u}_{l,2}) - J(\boldsymbol{m}, \boldsymbol{u}_l) & \ldots & J(\boldsymbol{m}, \mathbf{u}_{l,N_p}) - J(\boldsymbol{m}, \boldsymbol{u}_l) \end{bmatrix}^T. \tag{2.8}$$

If $d = N_p$, meaning the size of the perturbed ensemble is equal to the number of controls, the gradient $\nabla J(\boldsymbol{m}, \boldsymbol{u}_l)$ can directly be computed as:

$$\nabla J(\boldsymbol{m}, \boldsymbol{u}_l) = (\Delta \mathbf{U}_l)^{-1} \Delta \boldsymbol{j}_l. \tag{2.9}$$

However typically, the number of controls d is larger than the ensemble size N_p. This means the matrix $\Delta\mathbf{U}_l$ is singular and an inverse does not exist. In this case, an approximate gradient can still be computed as the underdetermined least squares solution:

$$\boldsymbol{g}_l = \nabla J(\boldsymbol{m}, \boldsymbol{u}_l) = \left(\Delta\mathbf{U}_l(\Delta\mathbf{U}_l)^T\right)^+ \Delta\mathbf{U}_l(\Delta\boldsymbol{j}_l), \tag{2.10}$$

where the + superscript indicates the Moore-Penrose pseudo-inverse and is computed using a singular value decomposition. (Strang (2006) [38]). Since only one realization is used, the notation is simplified by eliminating $\boldsymbol{m}$. The objective function value and gradient at $\boldsymbol{u}_l$ therefore become $J(\boldsymbol{u}_l)$ and $\nabla J(\boldsymbol{u}_l)$ respectively.

With the objective function J, control vector $\boldsymbol{u}$ and gradient vector $\boldsymbol{g}$ defined, lifecycle optimization can be treated as a constrained, continuous multi-variate optimization problem in d dimensions.

2.3.2 Steepest Ascent Algorithm

If $\boldsymbol{u}_k$ is the control vector at the k^{th} optimization iteration, the steepest ascent update can be defined as:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \alpha_k \boldsymbol{g}_k. \tag{2.11}$$

The step length parameter α_k is determined from line search. For steepest ascent update where Hessian approximation is not required, the Armijo back tracking line search can be used. See Nocedal & Wright (2006) [36] Section 3.1, p. 33 for the backtracking algorithm. $\boldsymbol{g}_k$ is the EnOpt gradient vector computed at the k^{th} iteration.

Due to its approximate nature the EnOpt gradient need not always be the direction of ascent and optimization can occasionally stagnate. This was observed experimentally. (figure 5.1a)

2.4 Exploiting An Ill-Posed Problem

While lifecycle optimization unquestionably forms the foundation of this thesis and has been covered in depth till now - the true objective of this thesis lies one step further.

Jansen et al. (2009) [25] observed that substantially different control strategies (arising from different control update frequencies) resulted in near identical NPV's. This result (amongst others in the paper) led to the conclusion that waterflood optimization was an ill-posed problem as there existed many more control variables than necessary. A similar effect was observed before in the minimization of a mismatch function for computer assisted history matching. (Oliver et al. (2008) [37]) Intuitively, an ill-posed problem is equivalent to a system of equations that contains more variables than equations and can consequently admit infinitely many solutions.

While this might appear as an issue, an ill-posed problem in the context of this thesis is actually quite favorable since it translates into potentially multiple control strategies which result in an overall NPV that is identical to or near the optimum but yet different enough to provide an operator with flexibility to respond to changing market conditions and financial positions. Flexible production strategies also provide the possibility of dealing with unforeseen maintenance activity such as well interventions which as mentioned in Van Essen et al. (2012) [41] is an issue that severely limits the practical usability of model reservoir based production inputs.

The primary objective of this thesis therefore becomes 'quantifying the ranges of feasible control strategies that lie within a user defined economic threshold.' To accomplish this objective it is necessary to explore the feasible region in the vicinity of the optimum control coordinate - a region where the objective function value does not decrease beyond the user defined threshold relative to the optimum. Refer to chapter 3 and 4 for details on the practical implementation.

Exploring the feasible region requires search directions. It is desirable to identify search directions which result in a minimal change in objective function value as this translates into longer exploration distances and potentially a greater range of feasible controls. The hierarchical multi-objective production optimization concept demonstrated by Van Essen et al. (2011) [40] and Fonseca (2011) [12] describes an approach to identify such directions. In brief, this approach requires computing the Hessian at the optimum and subsequently an SVD is applied to identify the Hessian's null space. The basis vectors that span this null space represent directions relative to the optimum coordinate along which the decrease in objective function value is slowest. Consequently these directions can be used for exploration. An extensive description is now presented:

Background: Hierarchical multi-objective optimization requires two objective functions with divergent goals. The primary objective focuses on long term (future) production by maximizing the cumulative undiscounted cash flow. The secondary objective focuses on (near initial) short term production by maximizing a highly discounted cumulative cash flow (where discount factor = 0.25). The two publications suggest that since lifecycle optimization is ill-posed, redundant degrees of freedom (DOF) exist even when the primary objective has reached optimality. This redundancy provides "maneuverability" in terms of control strategies and is used to improve the secondary objective.

To identify the redundant degrees of freedom Fonseca (2011) [12] suggest creating a quadratic approximation of the primary objective function via the (truncated) Taylor series expansion. The process is explained below:

Suppose $\boldsymbol{u}^*$ is the optimum coordinate and J is an objective function. Let $\Delta\boldsymbol{u}$ be a perturbation vector with the same length as $\boldsymbol{u}^*$ i.e., $d \times 1$. The Taylor series expansion around $\boldsymbol{u}^*$ can be expressed as:

$$J(\boldsymbol{u}^* + \Delta\boldsymbol{u}) \approx J(\boldsymbol{u}^*) + \nabla J(\boldsymbol{u}^*)^T \cdot \Delta\boldsymbol{u} + \frac{1}{2}\Delta\boldsymbol{u}^T \cdot \nabla^2 J(\boldsymbol{u}^*) \cdot \Delta\boldsymbol{u} + \mathcal{O}(\|\Delta\boldsymbol{u}\|^3). \tag{2.12}$$

Equation 2.12 is a quadratic approximation of the multivariate objective function J - an approximation which estimates local function behavior using only knowledge of the function's first and second derivatives. The term $\nabla J(\boldsymbol{u}^*)$ is the $d \times 1$ gradient vector while $\nabla^2 J(\boldsymbol{u}^*)$ is the $d \times d$ Hessian matrix. If $\boldsymbol{u}^*$ is the (local) optimum that lies in the interior of the feasible domain with no active constraints then,

$$\nabla J(\boldsymbol{u}^*) = \boldsymbol{0}. \tag{2.13}$$

Substituting 2.13 in equation 2.12 we obtain:

$$J(\boldsymbol{u}^* + \Delta\boldsymbol{u}) \approx J(\boldsymbol{u}^*) + \frac{1}{2}\Delta\boldsymbol{u}^T \cdot \nabla^2 J(\boldsymbol{u}^*) \cdot \Delta\boldsymbol{u} + \mathcal{O}(\|\Delta\boldsymbol{u}\|^3). \tag{2.14}$$

Equation 2.13 indicates that the gradient vanishes at the optimum. Consequently, the quadratic approximation in equation 2.14 is dependent only on the Hessian matrix.

Since $\boldsymbol{u}^*$ is the local maximum, the Hessian matrix $\boldsymbol{H} = \nabla^2 J(\boldsymbol{u}^*)$ at this coordinate is either:
1. negative definite - in which case the Hessian has full rank and redundant DOF's do not exist.
2. negative semi-definite - in which case the Hessian is rank deficient and both redundant DOF's and a null space exist.

Limiting consideration to the case of a negative semi-definite Hessian matrix, the null space of such a matrix $\boldsymbol{H} \in \mathbb{R}^{d\times d}$ consists of all vectors $\boldsymbol{x} \in \mathbb{R}^{d\times 1}$ for which $\boldsymbol{H}\boldsymbol{x} = 0$. This is in addition to the zero vector which is always a part of the null space.

The null space of the Hessian is identified by a singular value decomposition. (Strang (2006) [38]) As the Hessian is symmetric, the SVD becomes:

$$\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T. \tag{2.15}$$

Singular vectors $(\boldsymbol{q}_1, \ldots, \boldsymbol{q}_d)$ are contained in the columns of $\boldsymbol{Q}$ while the associated singular values $(\sigma_1, \ldots, \sigma_d)$ are contained along the diagonal of $\boldsymbol{\Lambda}$. The orthonormal basis $\boldsymbol{B}$ that defines the null space of $\boldsymbol{H}$ consists of the columns of $\boldsymbol{Q}$ corresponding to zero singular values in $\boldsymbol{\Lambda}$. This can be represented mathematically as:

$$\boldsymbol{B} \triangleq \big(\boldsymbol{q}_i \big| \sigma_i = 0,\ i = 1, \ldots, d\big). \tag{2.16}$$

If the perturbation vector $\Delta\boldsymbol{u}$ is selected from the null space of $\boldsymbol{H}$,

$$\nabla^2 J(\boldsymbol{u}^*) \cdot \Delta\boldsymbol{u} = \boldsymbol{0}. \tag{2.17}$$

Substituting 2.17 in equation 2.14 we obtain:

$$J(\boldsymbol{u}^* + \Delta\boldsymbol{u}) \approx J(\boldsymbol{u}^*) + \mathcal{O}(\|\Delta\boldsymbol{u}\|^3). \tag{2.18}$$

Equation 2.18 suggests that for a suitably selected $\Delta\boldsymbol{u}$ the adjusted control vector $(\boldsymbol{u}^* + \Delta\boldsymbol{u})$ produces a minimal change in objective function value relative to the optimum.

Van Essen et al. (2011) [40] and Fonseca (2011) [12] further discuss methods to project the improving search direction of the secondary objective (i.e., the gradient) onto the null space of the primary - resulting in both objectives being optimized - but that discussion is not of relevance to this thesis. Equation 2.18 does however

serve as justification to selecting the null space basis vectors of the Hessian as search directions.

In practice, the BFGS Hessian approximation is positive definite regardless of the true Hessian at a coordinate. (Nocedal & Wright (2006) [36] Section 6.3, p. 150) This means the Hessian matrix has full rank and no true null space exists. Van Essen et al. (2011) [40] and Fonseca (2011) [12] consequently suggest a relaxed definition for singular vectors to be admitted as basis vectors of the null space. An admission cut-off is defined based on relative singular values as:

$$\boldsymbol{B} \triangleq \left(\boldsymbol{q}_i \middle| \frac{\sigma_i}{\sigma_1} < 0.02,\ i = 1, \ldots, d \right), \tag{2.19}$$

where σ_1 is the largest singular value. This replaces the stricter requirement presented in equation 2.16. For the sake of a thorough exploration, this thesis sets the cut-off as $\sigma_i/\sigma_1 <= 1$ thereby ensuring all singular vectors are used as search directions.

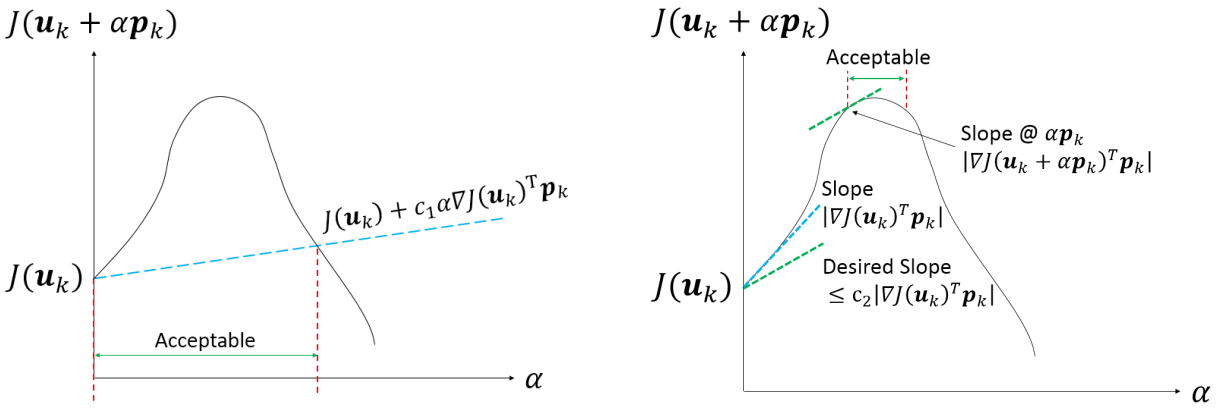
2.5 Approximating The Hessian Matrix

No reservoir simulator provides direct functionality to compute the Hessian. In this thesis the Broydon-Fletcher-Goldfarb-Shanno (BFGS) scheme is used to iteratively approximate the Hessian matrix. Iteratively building curvature/second order derivative information is computationally attractive in comparison to an approach such as finite difference Hessian approximation which requires $(d^2+3d)/2$ function evaluations where d is the number of controls. (See Dennis & Schnabel (1983) [7] p. 80, 105) This is especially true when function evaluations are expensive. Fonseca (2011) [12] also use BFGS Hessian approximation. Van Essen et al. (2011) [40] have access to an adjoint gradient and this justifies their choice of a finite difference Hessian.

2.5.1 Line Search

The BFGS scheme requires a line search which accounts for gradient information at each iterate to preserve Hessian positive definiteness. (Nocedal & Wright (2006) [36] Section 6.1, p. 143) For this reason, the strong Wolfe line search (SWLS) is used. The strong Wolfe conditions for a successful trial iterate $\boldsymbol{u}_{k+1}$ are:

$$\begin{aligned} J(\boldsymbol{u}_{k+1}) &\geqslant J(\boldsymbol{u}_k) + c_1 \alpha_k \nabla J(\boldsymbol{u}_k)^T \boldsymbol{p}_k \\ |\nabla J(\boldsymbol{u}_{k+1})^T \boldsymbol{p}_k| &\leqslant c_2 |\nabla J(\boldsymbol{u}_k)^T \boldsymbol{p}_k| \end{aligned} \tag{2.20}$$



(a) Sufficient Increase Condition

(b) Curvature Condition

Figure 2.1: Strong Wolfe conditions - line search is depicted as a univariate optimization problem in α where the goal is to find a step length suitable to both Wolfe conditions. In figure 2.1b the scalar product of gradient $\nabla J(\boldsymbol{u}_k)^T$ and search direction $\boldsymbol{p}_k$ indicates slope in the univariate sense - at $\alpha = 0$.

In the above inequalities, $\boldsymbol{p}_k$ is the search direction, α_k is the trial step length and $\nabla J(\boldsymbol{u}_k)^T \boldsymbol{p}_k$ is the scalar directional derivative of the gradient and search direction.

The 'sufficient increase' condition relates the minimum acceptable increase in objective function value to step length. c_1 is a constant set as 10^{-4}. Figure 2.1a indicates that a larger c_1 results in a stricter 'Sufficient Increase' requirement. The "curvature" condition indicates that at the trial iterate, the magnitude of the scalar product of the new gradient and search direction $|\nabla J(\boldsymbol{u}_{k+1})^T \boldsymbol{p}_k|$ should be smaller than that of the existing

gradient and search direction $c_2|\nabla J(\boldsymbol{u}_k)^T \boldsymbol{p}_k|$ - by a certain margin. This condition forces α to lie in the neighborhood of the local minimizer or stationary point of $J(\boldsymbol{u}_k + \alpha \boldsymbol{p}_k)$. The value of c_2 can be selected in the range: $0 < c_1 < c_2 < 1$ and is set as 0.9 in the experiments.

From the perspective of an EnOpt gradient, it should be noted that testing for the "curvature" condition is computationally expensive since the gradient has to be calculated at each intermediate test iterate of line search - within one optimization iteration. This condition does however ensure that the gradients are sampled at points which allow the BFGS scheme to appropriately capture curvature information - allowing for a better Hessian approximation. (Nocedal & Wright (2006) [36] Section 6.1, p. 142)

The Armijo backtracking line search (ABTLS) decreases step length (backtracks) relative to a trial step length till the 'sufficient increase' condition is met. However there is no provision to test the gradient at any intermediate iterate to see if the "curvature" condition is satisfied - which therefore makes it unsuitable for BFGS Hessian approximation. (Nocedal & Wright (2006) [36] Section 6.1, p. 143)

Refer to appendix A1.2 for details on how the trial step length is calculated. With a known trial step length, the step length bracketing and zoom operations suggested in Nocedal & Wright (2006) [36] Algorithm 3.5, 3.6, p. 60, 61 are used to identify a final step length that satisfies both strong Wolfe conditions - if it exists. The MATLAB implementation of these algorithms sourced from Guipeng Li [29] is used.

2.5.2 Steepest Ascent Algorithm with BFGS

The steepest ascent update with BFGS approximation consists of the following steps:

1. Set the search direction $\boldsymbol{p}_k$ where,
   (a) $\boldsymbol{p}_k = \nabla J(\boldsymbol{u}_k)$ - steepest ascent
   (b) $\boldsymbol{p}_k = \boldsymbol{H}_k^{-1} \nabla J(\boldsymbol{u}_k)$ - quasi-Newton (not used)
2. Update controls: $\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \alpha_k \boldsymbol{p}_k$ Where α_k is found through strong Wolfe line search
3. Set $\boldsymbol{s}_k = \boldsymbol{u}_{k+1} - \boldsymbol{u}_k$
4. Set $\boldsymbol{y}_k = \nabla J(\boldsymbol{u}_k) - \nabla J(\boldsymbol{u}_{k+1})$
5. The updated approximate Hessian is: $\boldsymbol{H}_{k+1} = \boldsymbol{H}_k - \dfrac{\boldsymbol{H}_k \boldsymbol{s}_k \boldsymbol{s}_k^T \boldsymbol{H}_k}{\boldsymbol{s}_k^T \boldsymbol{H}_k \boldsymbol{s}_k} + \dfrac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}$

Note that Nocedal & Wright (2006) [36] use $\boldsymbol{B}_k$ to refer to the approximate Hessian and $\boldsymbol{H}_k$ to refer to its inverse. This thesis uses $\boldsymbol{H}_k$ to refer to the approximate Hessian instead. $\boldsymbol{s}_k$ represents the difference between two consecutive iterates while $\boldsymbol{y}_k$ represents the difference in gradient at consecutive iterates. At the first iteration, the Hessian is set as the identity matrix.

Typically $\boldsymbol{s}_k = \alpha_k \boldsymbol{p}_k$ is used and is acceptable for unconstrained optimization problems. When simple bound constraints are used, as done in this thesis $\alpha_k \boldsymbol{p}_k$ need not equal $\boldsymbol{u}_{k+1} - \boldsymbol{u}_k$. $\alpha_k \boldsymbol{p}_k$ represents the suggested change in controls whereas $\boldsymbol{u}_{k+1} - \boldsymbol{u}_k$ is the actual change. The difference arises when controls in $\boldsymbol{u}_k$ are situated at bounds and the direction vector $\boldsymbol{p}_k$ points outwards - meaning it attempts to decrease a control at the lower bound or increase a control at the upper bound. Controls that exceed bounds during a line search update have to be reset to their minimum/maximum values and this gives rise to the discrepancy.

Gill et al. (2001) [18] mention that in quasi-Newton methods such as BFGS, the second order derivative information is accumulated in a sequence of expanding subspaces. They state, "at the k^{th} iteration ($k < d$), function curvature is known along vectors that lie in a gradient subspace that can have at most $k + 1$ dimensions." (d is the number of controls to be optimized) This result implies for an approximate Hessian to completely capture curvature - at least d iterations of the BFGS update are required. A BFGS Hessian update counts as valid only when both 'sufficient increase' and "curvature" conditions of the strong Wolfe line search are met.

To ensure sufficient BFGS updates before convergence, the steepest ascent search direction is used in place of the quasi-Newton search direction. Griffin (2012) [22] present the BFGS algorithm from a maximization point of view and this publication was useful in solving issues related to sign convention. The BFGS scheme has been validated on both the regular (minimization - appendix A1.3.3) and inverted (maximization - appendix A1.3.4) 2D Rosenbrock function.

2.6 Reservoir model

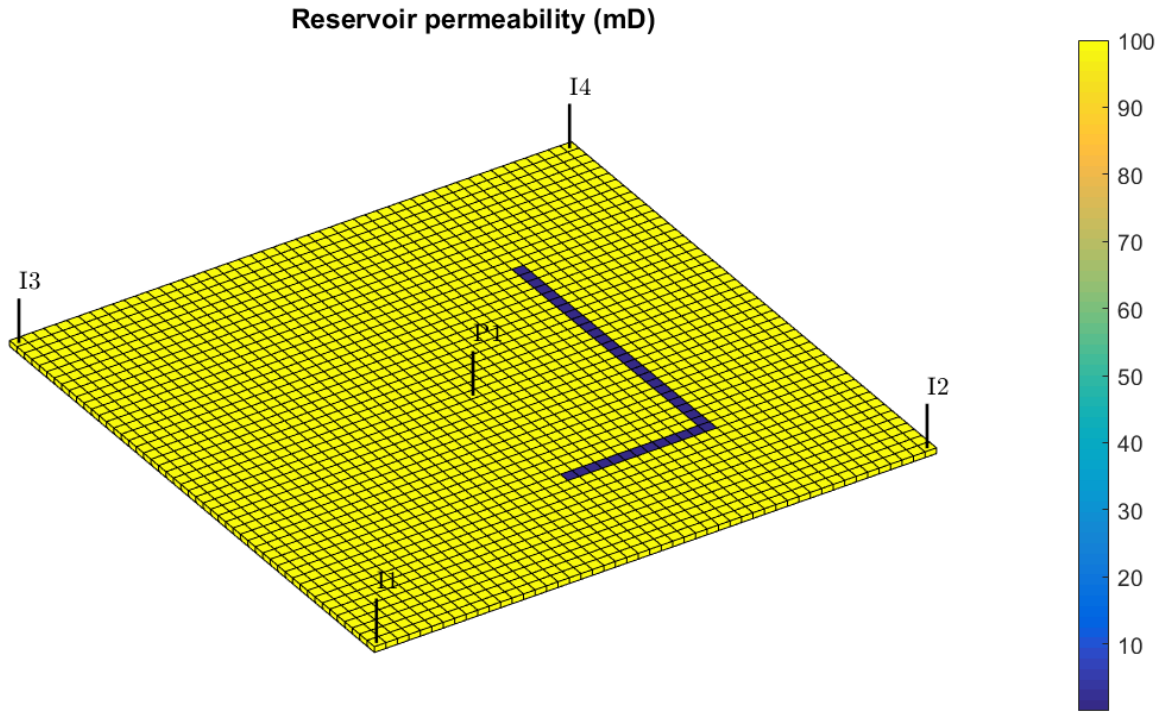


Figure 2.2: Reservoir permeability

This thesis uses the MATLAB Reservoir Simulation Toolbox (MRST) by Sintef [30]. Four injector wells equipped with one inflow control valve (ICV) per well are specified. From a reservoir simulation perspective, each ICV setting is modelled as the well productivity index that indicates connectivity between a well and its reservoir gridblock. The aperture for each ICV can be set as a scalar value between 10^{-6} and 1. The production period is 4000 days divided in 10 time steps of 400 days each. Using equation 2.3 this results in an optimization problem consisting of 40 variables. (4 wells $\times$ 1 ICV/well $\times$ 10 time steps)

Properties of the reservoir model are as follows:

Grid blocks	$49 \times 49 \times 1$
Reservoir permeability	100 mD
Fault permeability	0 mD
Porosity	0.20
Initial S_w	0.20
Initial P_{res}	200 bar

Table 2.2: Initial reservoir properties

The reservoir consists of a single layer with 2401 gridblocks and has an 'L' shaped fault that impedes fluid flow - posing a challenge from a lifecycle optimization perspective. Two phase incompressible flow is considered where the effects of gravity and capillary pressure are neglected. The four injectors are operated at a constant BHP of 300 bars while the producer is operated at a constant BHP of 100 bars.

Property	Water	Oil	Unit
Viscosity	0.4	0.9	cP
Density	1014	859	kg/m^3
Relperm exponent	2	2	[-]
Phase residual saturation	0.2	0.3	[-]
Phase relperm @ residual saturation	0.4	0.8	[-]

Table 2.3: Fluid properties

The discounted net present value is calculated based on equation 2.1 and the following inputs:

Oil price	150 $\$/m^3$
Water production cost	25 $\$/m^3$
Water injection cost	5 $\$/m^3$
Discount factor	0.08

Table 2.4: NPV calculation parameters

Chapter 3

Exploration

The outcome of optimization is a control strategy that results in a (potentially local) maximum NPV for the previously described reservoir model. It is now useful to explore the d-dimensional control space for strategies that result in near optimal NPV. In the context of exploration, 'near optimal' refers to an acceptable decrement in objective function value - relative to the optimum. This user specifiable 'acceptable decrement' defines a lower threshold $J_{\min}$ for the objective function value where associated control strategies can still be called feasible. The lesser the threshold, the larger the range of strategies admitted in the feasible region. The goal of exploration is thus identifying the shape of this feasible region - which is a d-dimensional space containing all feasible control strategies - including the optimum.

Mathematically, the feasible region $\boldsymbol{F}$ consists of the set of all control vectors $\boldsymbol{u}$ in $\mathbb{R}^d$ for which the function value condition $J(\boldsymbol{u}) \geqslant J_{\min}$ holds.

$$\boldsymbol{F} = \{\boldsymbol{u} \in \mathbb{R}^d \mid J(\boldsymbol{u}) \geqslant J_{\min}\} \tag{3.1}$$

Identifying the shape of the feasible region is equivalent to identifying the shape of the objective function contour/boundary/level-set for a given function value threshold. A thorough exploration requires many function evaluations and translates to a large number of samples on this boundary.

The samples/points found on the boundary of the feasible region serve as vertices to a d-dimensional polytope which constrains the centre and axis lengths of an inscribed hyper-ellipsoid. (chapter 4) More points on the function boundary results in more polytope constraints and consequently a better ellipsoidal approximation of the feasible region. Exploration however is subject to the curse of dimensionality meaning the number of function evaluations needed to map the boundary increases exponentially with number of dimensions. So, while it may be computationally intractable to map the function boundary perfectly a coarse approximation can still be made.

Regarding terminology, this thesis uses the terms 'singular vector', 'exploration direction' and 'direction vector' interchangeably.

3.1 In practice

The singular value decomposition of the BFGS Hessian at the optimum gives d orthonormal singular vectors $(\boldsymbol{q}_1, \ldots, \boldsymbol{q}_d)$. Exploration is performed by varying the optimum coordinate along the positive and negative extents of only one singular vector at a time and this results in $2 \times d$ points obtained on the boundary of the feasible region. The algorithm for exploration is presented in 3.2. Similar one dimensional exploration is performed in Director & Hatchel (1977) [9] and Zamora-Sillero et al. (2011) [44].

It should be noted that exploration along only d singular vectors means function value information away from the exploration axis (defined by singular vectors) is not available. This means - with the current exploration algorithm - it is unknown how the objective function value changes/evolves when the optimum coordinate is perturbed along say, two or more singular vectors simultaneously.

If the optimum coordinate were to be varied along two singular vector directions simultaneously, then function value information is identified along a 2-dimensional plane in d-dimensional control vector space. This idea has been used only to diagnose the quality of ellipsoid fit in the feasible region - in an approach this thesis terms '2D Plane Scan'. While such exploration does indeed lead to a larger number of points on the feasible region boundary - which is desirable - the main issue which disqualifies its consideration is computational cost. For example: Two singular vectors can be chosen from d total vectors in ${}^dC_2$ combinations. In the 40 dimension reservoir case, this results in ${}^{40}C_2 = 780$ unique 2-dimensional planes that require exploration.

$$
\begin{aligned}
&\boldsymbol{u} \leftarrow \boldsymbol{u}_{\text{optimum}} \\
&\alpha \leftarrow \alpha_{\text{init}} \\
&\textbf{for } i \in \{1, \cdots, d\} \\
&\quad \textbf{while } \alpha > \alpha_{min} \\
&\quad\quad \boldsymbol{u}_{\text{temp}} \leftarrow \boldsymbol{u} + \alpha \boldsymbol{q}_i \\
&\quad\quad \textbf{Evaluate } J(\boldsymbol{u}_{\text{temp}}) \\
&\quad\quad \textbf{if } J(\boldsymbol{u}_{\text{temp}}) < J_{\text{min}} \\
&\quad\quad\quad \alpha \leftarrow \alpha \times 0.5 \\
&\quad\quad \textbf{else} \\
&\quad\quad\quad \textbf{Store } \left(J(\boldsymbol{u}_{\text{temp}}), \boldsymbol{u}_{\text{temp}}\right) \\
&\quad\quad\quad \boldsymbol{u} \leftarrow \boldsymbol{u}_{\text{temp}} \\
&\quad\quad \text{end } \textbf{if} \\
&\quad \text{end } \textbf{while} \\
&\quad \text{Reset } \boldsymbol{u} \leftarrow \boldsymbol{u}_{\text{optimum}}, \alpha \leftarrow \alpha_{\text{init}} \\
&\quad \text{Repeat } \textbf{while} \text{ block with } -\boldsymbol{q}_i \\
&\text{end } \textbf{for}
\end{aligned}
\tag{3.2}
$$

The above algorithm can be interpreted as a 1D line search where singular vectors are used as search directions and termination is based on a minimum acceptable step size condition. Vector $\boldsymbol{q}_i$ represents the i^{th} singular vector. The function values and coordinates at each intermediate step are stored and this allows visualizing the change in objective function value relative to distance from the optimum. From a parallel computation point of view, the $2 \times d$ line searches can be initiated simultaneously (along positive and negative extents of d singular vectors).

3.2 Log Transformed Controls

As indicated in equation 2.2, lifecycle optimization is a constrained optimization problem. The vectors $\boldsymbol{u}^{min}$ and $\boldsymbol{u}^{max}$ indicate the lower and upper bounds for values the ICV settings can take. The presence of bounds poses a challenge for exploration.

3.2.1 The Issue With Projection

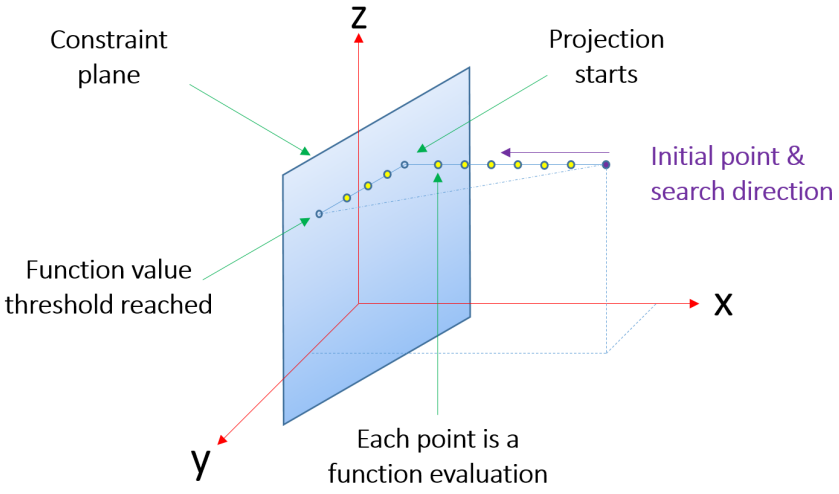


Figure 3.1: Bound constraints are a hindrance to exploration

Experimentally it was observed that the optimum control vector has many controls at bounds. A control at the lower bound for example indicates that the optimizer has deemed zero injection for the associated well and time step as ideal from an NPV standpoint.

If an exploration direction has a vector element that results in an increase in the magnitude of a control at the lower bound then this is not a problem. However, if the direction vector element points outwards meaning it tries to decrease a control already at the lower bound or say, tries to increase a control at the upper bound then exploration is possible only with 'projection' - where the control violating the bound constraint is projected back to its minimum or maximum value - at each exploration step.

Points found through projection-based exploration do not lie in the span of the singular vectors used for exploration. This means the vector distance from the exploration origin (i.e., the optimum coordinate) to a projected point on the feasible region boundary cannot be expressed in terms of the singular vector used for exploration. This however is an issue specific to the ellipsoid fitting workflow described in chapter 4. Projection also means the exploration directions are no longer mutually orthogonal.

These issues are avoided by applying a log transformation to the controls.

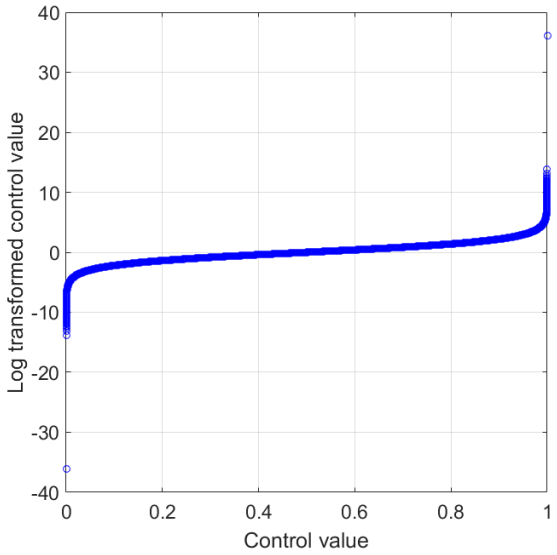


Figure 3.2: Log transformed vs. Regular controls

The log transformation is a non-linear transform which maps the lower bound $u_i^{min} = 10^{-6}$ to $s_i = -\infty$ and the upper bound $u_i^{max} = 1$ to $s_i = +\infty$ where $i = 1, \cdots, d$ and s_i is a control in the log transformed space. (Gao et al. (2006) [17]) As the lower/upper bounds are eliminated the log transformed space is unconstrained and consequently optimization, exploration and ellipsoid fitting are performed in this space.

The i^{th} control u_i is converted to the log transformed space as:

$$s_i = \ln\left(\frac{u_i - u_i^{min}}{u_i^{max} - u_i}\right) \tag{3.3}$$

To return to the regular space the inverse transform is defined as:

$$u_i = \frac{\exp(s_i)u_i^{max} + u_i^{min}}{1 + \exp(s_i)} \tag{3.4}$$

The EnOpt gradient & BFGS Hessian are computed with respect to the log transformed coordinates and the SVD of the Hessian still provides the exploration directions in the log transformed space. The regular space is used only to evaluate the objective function value and generate the ensemble of perturbed control vectors for EnOpt gradient computation. Issues relating to the control vector perturbation process - as described in Wang et al. (2007) [42] - are addressed in appendix A1.4. This appendix section also describes the gradient computation workflow and presents the motivation for restricting controls in the control vector ensemble more conservatively to within $u_{i,\text{perturb}}^{min} = 10^{-4}$ and $u_{i,\text{perturb}}^{max} = 0.9999$.

In practice numerical issues arise with equation 3.3 when $u_i = 10^{-6}$ or $u_i = 1$ as the corresponding log transformed controls reach $-\inf$ and $+\inf$ respectively. This is addressed locally in the MATLAB code by adding `+eps` to all vector elements in $\boldsymbol{u}^{max}$ and `-eps` to all vector elements in $\boldsymbol{u}^{min}$ where `eps` $= 2.2204 \times 10^{-16}$. Consequently for $u_i = 10^{-6}$, $s_i = -36.0437$ and for $u_i = 1$, $s_i = +36.0437$ as can be seen in figure 3.2.

3.3 2D Plane Scan

Exploration can be performed by varying the optimum control vector along two singular vector directions simultaneously. This way, the function value information is identified along a 2-dimensional plane in d-dimensional control vector space. The algorithm for this exploration is presented below and consists of two steps:

Step 1 (Primary exploration):

$$
\begin{aligned}
&i \leftarrow 1\\
&\alpha \leftarrow \alpha_{\text{init}}\\
&\boldsymbol{u} \leftarrow \boldsymbol{u}_{\text{optimum}}\\
&\boldsymbol{U}(:, i) \leftarrow \boldsymbol{u}\\
&\textbf{while } \alpha > \alpha_{min}\\
&\quad \boldsymbol{u}_{\text{temp}}^{\text{primary}} \leftarrow \boldsymbol{u} + \alpha \boldsymbol{p}\\
&\quad \textbf{Evaluate } J(\boldsymbol{u}_{\text{temp}}^{\text{primary}})\\
&\quad \textbf{if } J(\boldsymbol{u}_{\text{temp}}^{\text{primary}}) < J_{\text{min}}\\
&\quad\quad \alpha \leftarrow \alpha_{min}\\
&\quad \textbf{else}\\
&\quad\quad i \leftarrow i + 1\\
&\quad\quad \textbf{Store } J(\boldsymbol{u}_{\text{temp}}^{\text{primary}})\\
&\quad\quad \boldsymbol{u} \leftarrow \boldsymbol{u}_{\text{temp}}^{\text{primary}}\\
&\quad\quad \boldsymbol{U}(:, i) \leftarrow \boldsymbol{u}\\
&\quad \text{end } \textbf{if}\\
&\text{end } \textbf{while}\\
&\text{Reset } \boldsymbol{u} \leftarrow \boldsymbol{u}_{\text{optimum}}, \alpha \leftarrow \alpha_{\text{init}}\\
&\text{Repeat } \textbf{while} \text{ block with } -\boldsymbol{p}
\end{aligned}
\tag{3.5}
$$

Step 2 (Secondary exploration):

$$
\begin{aligned}
&\alpha \leftarrow \alpha_{\text{init}}\\
&\textbf{for } j \in \{1, \cdots, i\}\\
&\quad \boldsymbol{u} \leftarrow \boldsymbol{U}(:, j)\\
&\quad \textbf{while } \alpha > \alpha_{min}\\
&\quad\quad \boldsymbol{u}_{\text{temp}}^{\text{secondary}} \leftarrow \boldsymbol{u} + \alpha \boldsymbol{s}\\
&\quad\quad \textbf{Evaluate } J(\boldsymbol{u}_{\text{temp}}^{\text{secondary}})\\
&\quad\quad \textbf{if } J(\boldsymbol{u}_{\text{temp}}^{\text{secondary}}) < J_{\text{min}}\\
&\quad\quad\quad \alpha \leftarrow \alpha_{min}\\
&\quad\quad \textbf{else}\\
&\quad\quad\quad \textbf{Store } \left(J(\boldsymbol{u}_{\text{temp}}^{\text{secondary}}), \boldsymbol{u}_{\text{temp}}^{\text{secondary}}\right)\\
&\quad\quad\quad \boldsymbol{u} \leftarrow \boldsymbol{u}_{\text{temp}}^{\text{secondary}}\\
&\quad\quad \text{end } \textbf{if}\\
&\quad \text{end } \textbf{while}\\
&\quad \text{Reset } \boldsymbol{u} \leftarrow \boldsymbol{U}(:, j), \alpha \leftarrow \alpha_{\text{init}}\\
&\quad \text{Repeat } \textbf{while} \text{ block with } -\boldsymbol{s}\\
&\text{end } \textbf{for}
\end{aligned}
\tag{3.6}
$$

In step 1, a primary direction $\boldsymbol{p}$ is selected from d total singular vectors and a line search is performed along its positive and negative extents. The control vector coordinates at each primary exploration step are stored as column vectors in a matrix $\boldsymbol{U}$. These then serve as the initial coordinates for exploration along the secondary direction $\boldsymbol{s}$ in step 2, where $\boldsymbol{s}$ is chosen from the remaining $d-1$ singular vectors.

From a parallel computation perspective, the i feasible control vector coordinates found at the end of primary exploration are used to initiate $2 \times i$ parallel line searches for the secondary exploration phase. $2 \times i$ arises from the fact that both positive and negative extents of $\boldsymbol{s}$ need to be explored.

To visualize the 2D plane (that exists in d-dimensional space), the number of steps along the positive and negative extents of each primary and secondary exploration iteration are stored. With the step length α known, this information is used to create a proxy 2D plot where the function value information is presented in terms of marker color. The 2D plane scan results are presented in section 5.3.

To be clear in both plane scan algorithms 3.5 and 3.6, step size contraction is not used. Instead, a constant initial step size is specified ($\alpha_{\text{init}} = 2.5$ in experiments) and if it is observed that function value at an exploration step decreases below the acceptable threshold, exploration along that extent is simply terminated.

Chapter 4

Approximating the feasible region

4.1 Workflow overview

The goal of this thesis is to capture the feasible region where the objective function value is greater than some threshold. The outcome of exploration along positive and negative extents of each singular vector is a set of $2 \times d$ extremapoints at the boundary of the feasible region. See Figure 4.1. While sampling $2 \times d$ points in d-dimensional space is not thorough by any means, it still very coarsely captures the shape of the objective function contour.

With orthogonal exploration directions, the extremapoints found lie on linearly independent singular vectors and serve as vertices to a d-dimensional polytope (d-polytope). A polytope is a convex bounded polyhedron. See theorem 2 in [43]. While a polytope is itself an approximation of the feasible region, this thesis explores the idea of using an ellipsoid inscribed within such a polytope.

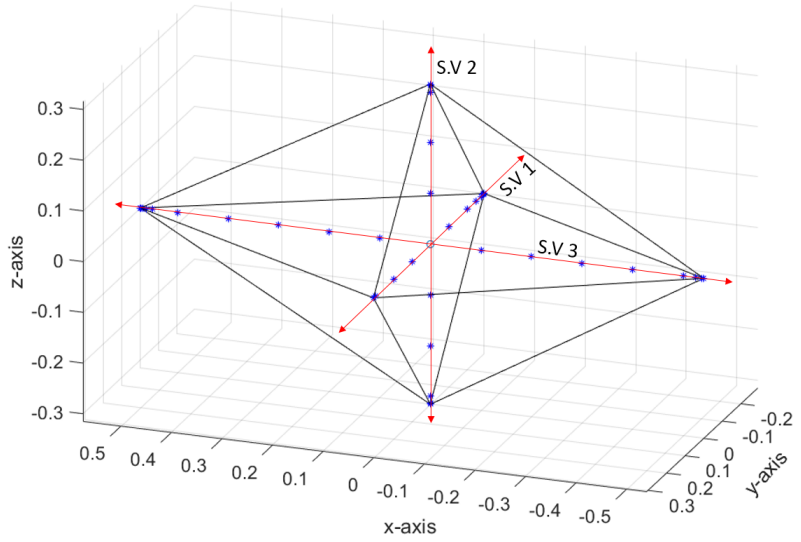


Figure 4.1: This 3-polytope is contained in the feasible region of the 3D Rosenbrock function. The optimum is chosen as the coordinate (0,0,0) for easy visualization. Singular vectors from the Hessian's SVD at this coordinate are used as search directions - which happen to be parallel to the Cartesian axes. The intermediate steps in line search are shown in blue. Exploration is terminated when the function value at a point exceeds a predefined threshold and the point immediately prior is termed an extremapoint. With 3 singular vectors, 6 extremapoints are obtained.

Inscribing an ellipsoid is treated here as an optimization problem. See section 8.4.2 in [2] and appendix A2.4 for more information. Convex optimization requires constraints that help define the centre, orientations and axis lengths of an ellipsoid. In this thesis, the d singular vector directions from the Hessian's SVD serve as axis orientations for the inscribed ellipsoid. This serves as a simplifying assumption and leaves only the centre and axis lengths to be found through optimization. In fact if necessary, the centre of the inscribed ellipsoid can also be selected beforehand (as the optimum) leaving only the axis lengths to be found through convex optimization. The disadvantage with both these approaches is that the quality of the BFGS Hessian approximation now influences the orientations of the inscribed ellipsoid.

Every polytope can be represented in two equivalent forms:

1. As the convex hull of its vertices (Figure 4.2)
2. As the set of solutions to a system of linear inequalities where each inequality defines a unique facet of the polytope (Figure 4.3). In this sense, $\boldsymbol{C} = \{\boldsymbol{x} \mid \boldsymbol{a}_i\boldsymbol{x} \leqslant b_i,\ i = 1, \ldots, k\}$ is a polytope that consists of the set of all elements $\boldsymbol{x}$ for which the k linear inequalities hold.

These are the vertex and halfspace representations respectively. For additional details refer to Glossary 15.1 in [24] & Theorem 2.49 in [45]. A facet is a $d-1$ dimensional face of the polytope. These inequalities or polytope facets serve as constraints to the convex optimization problem. See appendix A2.4. The linear inequalities that define these facets can be generated from the vertices (extremapoints) through a process known as facet enumeration. The `vert2con` package from MATLAB file exchange [32] is used for this.

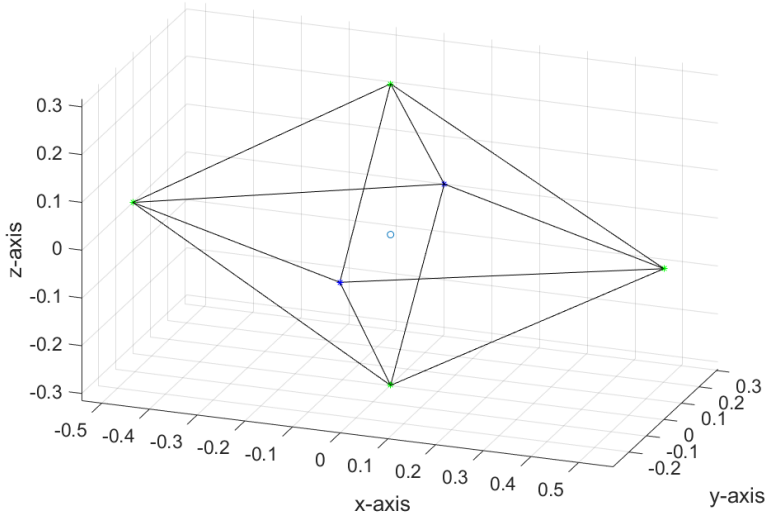


Figure 4.2: The six vertices of the 3-polytope

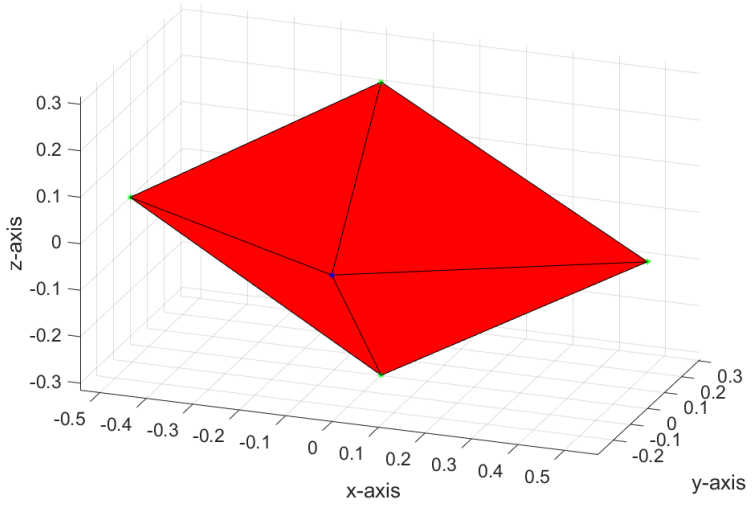


Figure 4.3: The eight facets of the 3-polytope are marked in red

If all $2 \times d$ extremapoints are used at once as vertices of the polytope, the number of constraints (facets) generated becomes intractable (e.g., 40 dimensions = 80 points $(2 \times d) = 2^{40}$ constraints). This is because the number of constraints scales exponentially in relation to the number of dimensions. To keep the optimization problem realistic, this thesis replaces the single d-polytope with multiple lower dimensional m-polytopes. See figure 4.4. Each m-polytope is defined in a unique m-dimensional space (m-space for short). Since exploration is performed away from the optimum, the optimum itself serves as the origin or zero coordinate in each m-dimensional space. m orthonormal singular vectors replace the Cartesian axes as basis vectors in each such space. With m singular vectors, only the subset of extremapoints in d-dimensional space that lie in the span of these m vectors become vertices to an m-polytope. The singular vector coordinate system in each m-space converts the coordinates of the extremapoints originally in d-dimensions into coordinates in m-dimensions. Refer to appendix A2.1 for more information. This is possible because the distance from the exploration origin (optimum) to each extremapoint can be expressed as some linear combination of the singular vectors.

This way, a set of vertices can be defined for each m-polytope. These sets of vertices are used to enumerate the facets of each m-polytope. m dimensions can be chosen from d total dimensions in ${}^{d}C_{m}$ unique combinations:

$$ {}^{d}C_{m} = \frac{d!}{m!(d-m)!} \tag{4.1} $$

The facets of the ${}^{d}C_{m}$ m-polytopes now replace facets of the single d-polytope as constraints in the optimization formulation. See appendix A2.4 for more information.

It should be noted while each m-space consists of a unique combination of singular vectors, a given singular vector can itself be included in multiple m-spaces. This means the extremapoints in d-dimensional space that lie in the span of a singular vector will become part of each m-polytope and m-space that incorporates this singular vector as basis. See underlined coordinates in Figure 4.4b & 4.4d where $d = 3$ and $m = 2$.

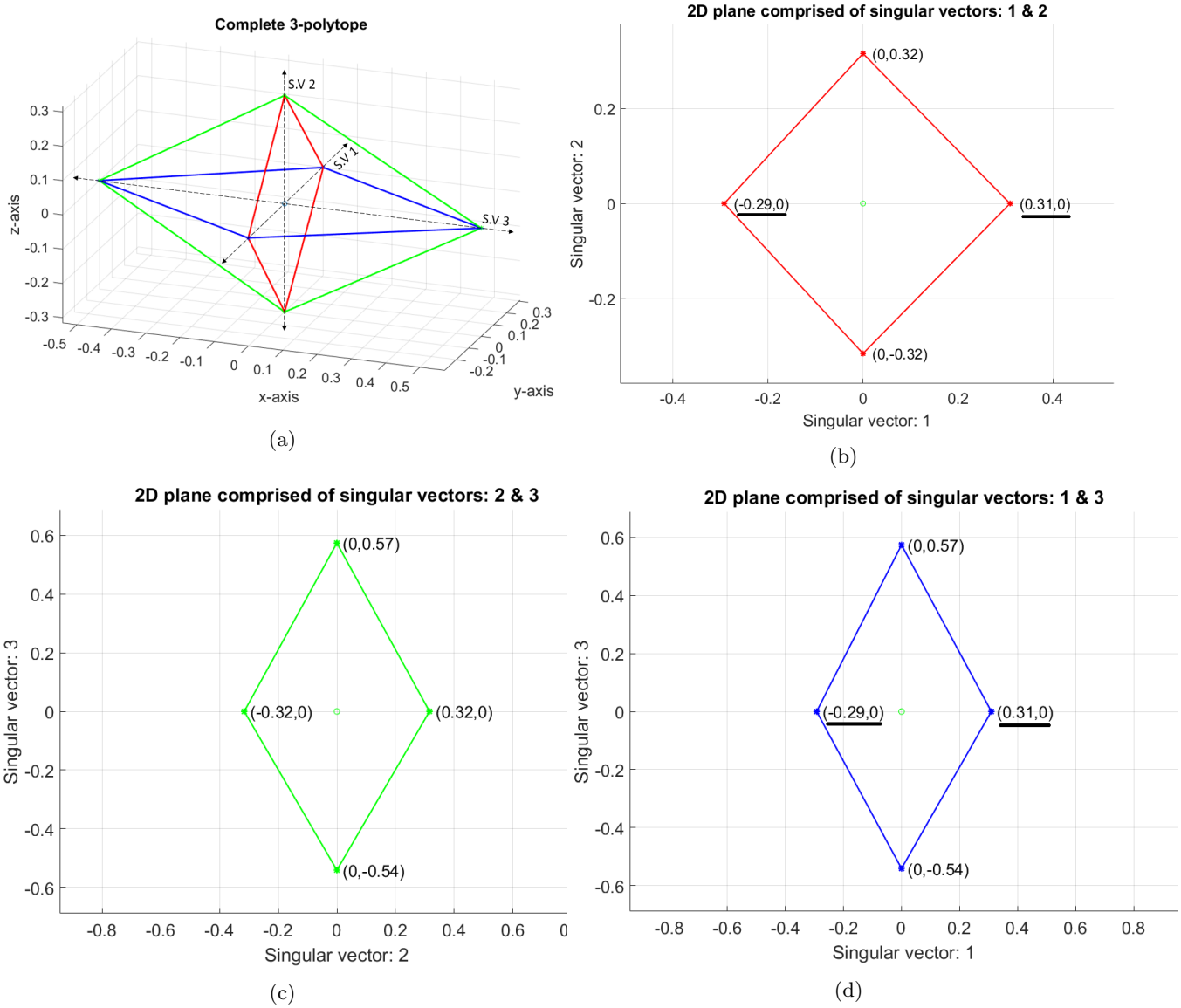


Figure 4.4: In this figure the 3-polytope is replaced by ${}^{3}C_{2} = 3$ 2-polytopes. The dashed lines in 4.4a represent the 1D span of each singular vector. In the remaining figures, the small green circle at coordinate (0,0) represents the origin of each 2D plane. Each 2-polytope is defined in a two dimensional space and consists of a subset of the vertices of the complete 3-polytope. The singular vectors replace the Cartesian axes coordinate system in each 2D space. They also allow the coordinates in 3D space to be represented in 2D. The facets (edges) of the three 2-polytopes replace the facets of the 3-polytope in Figure 4.3 as constraints in the optimization formulation.

Each m-polytope constrains an m-dimensional cross-section of the complete d-dimensional ellipsoid. An m-dimensional cross-section is a unique m-dimensional ellipsoid (or m-ellipsoid for short) that is comprised of a subset of the axes of the complete d-ellipsoid. Recall from earlier that the complete d-ellipsoid uses the singular vector directions as axes orientations. This way each singular vector is paired with one d-ellipsoid axis. This results in two properties: (See Figure 4.5)

1. Each m-ellipsoid has axes orientation parallel to the singular vector basis/coordinate system of its m-space.
2. The singular vectors that define an m-space determine the subset of d-ellipsoid axes that are included in the inscribed m-ellipsoid.

As with singular vectors and m-spaces, each axis of the d-ellipsoid will be included in multiple m-ellipsoids. (See Figure 4.5, where each axis of the 3-ellipsoid is included in two 2-ellipsoids) The centre and axis lengths of each m-ellipsoid are selected as the appropriate subset of components of a common/shared $d \times 1$ centre vector and $d \times d$ diagonal axis length matrix - See appendix A2.2 and algorithm A2.19. This way, each m-ellipsoid that includes a specific axis of the d-ellipsoid will have both an identical axis length and common centre coordinate along the paired singular vector basis. Note that by this definition each m-polytope constrains a unique subset of the previously mentioned centre vector and axis length matrix - specifically the subset that defines the associated m-ellipsoid. Convex optimization [19] [20] is subsequently used to determine the ideal $d \times 1$ centre vector and $d \times d$ diagonal axis length matrix that ensures the ${}^{d}C_{m}$ m-ellipsoids fit in their corresponding m-polytopes.

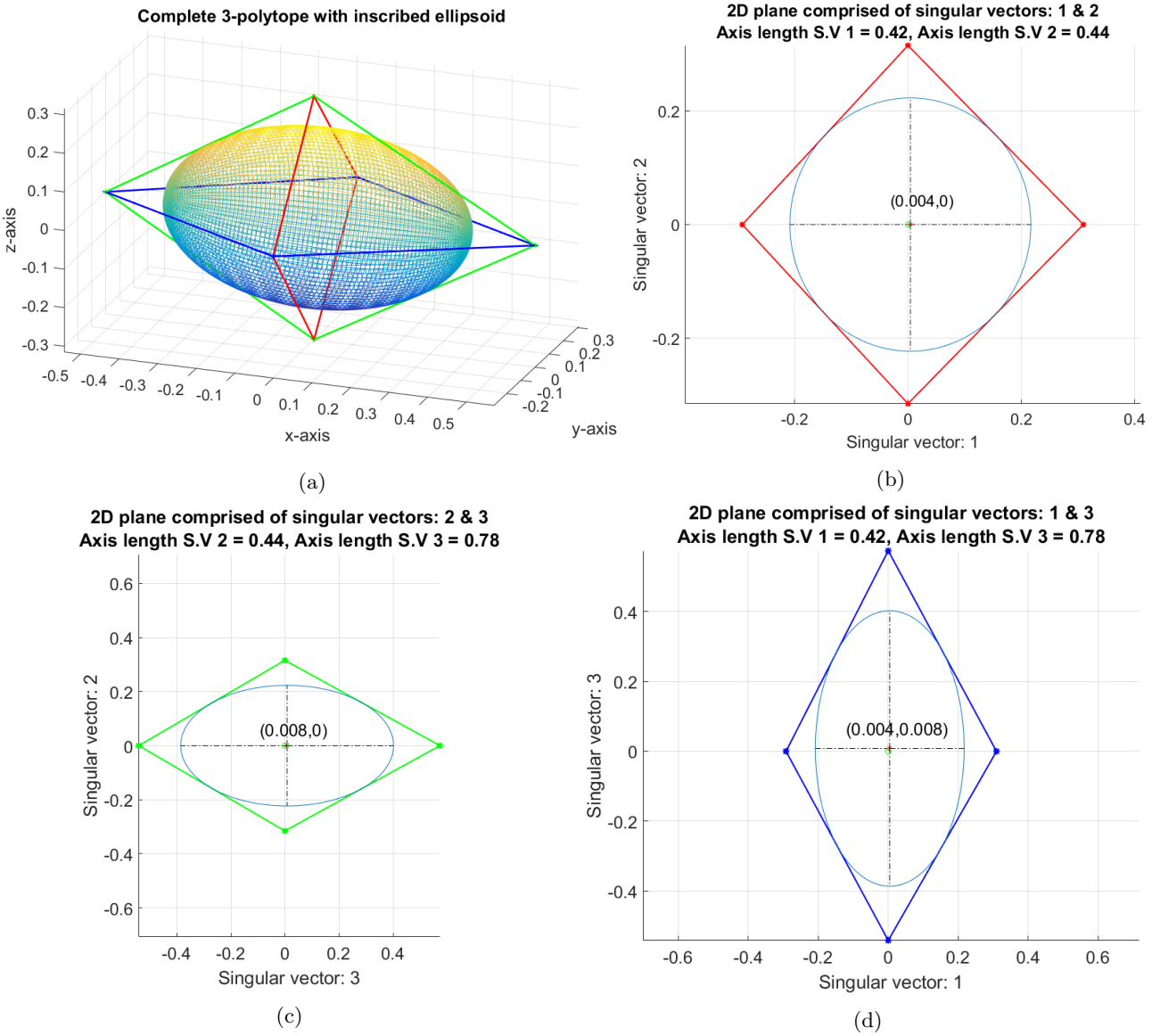


Figure 4.5: Each 2-polytope constrains a 2-dimensional cross-section of the complete 3-ellipsoid. The red point at the intersection of the ellipse axes indicates the ellipse centre whose coordinates are annotated. Each 2-ellipsoid has axes orientation parallel to the singular vector coordinate system.

The figures 4.5b - 4.5d represent the largest 2-ellipsoids that can be inscribed within the three 2-polytope constraints. The axis of the 3-ellipsoid paired with SV1 is included in two 2-ellipsoids - in Figure 4.5b & 4.5d respectively. This axis has an identical length and centre coordinate in both 2-ellipsoids. The remaining axes of the 3-ellipsoid paired with SV2 and SV3 also follow this rule. The CVX convex optimization package for MATLAB [19] [20] is used to inscribe each 2-ellipsoid. (The ellipsoids are plotted with [35])

These definitions become necessary since the end goal is inscribing an ellipsoid in d-dimensions. This means the m-dimensional cross-sections of the d-ellipsoid cannot be treated in isolation.

Since optimization ensures that each ellipsoid cross-section (m-ellipsoid) is inscribed within its m-polytope constraint, the reconstructed d-dimensional ellipsoid will also remain within each of these ${}^{d}C_{m}$ m-polytope constraints. This is seen in Figure 4.5a where the reconstructed 3-ellipsoid remains completely within the three 2-polytope constraints.

The reconstructed d-dimensional ellipsoid's surface replaces the facets of the d-polytope as the boundary of the approximated feasible region. Consequently this ellipsoid (defined by a $d \times d$ matrix and $d \times 1$ centre vector) becomes a more compact representation of the feasible region at the cost of being a poorer approximation. See appendix A2.3 for a mathematical description of an ellipsoid. The final reconstructed d-ellipsoid is henceforth referred to as d-CSC-MVIE which stands for d-dimensional cross-section constrained maximum volume inscribed ellipsoid.

The steps involved in the ellipsoid fitting workflow can be summarized as follows:

1. Split d singular vectors into ${}^{d}C_{m}$ groups of m vectors where each group defines a unique m-space.
2. Convert the extremapoints originally in d-dimensions to m-polytope vertices in each m-space.
3. Use the m-polytope vertices (in each m-dimensional space) to enumerate facets using a MATLAB package such as `vert2con` [32].
4. Using these facets as constraints, solve ${}^{d}C_{m}$ convex optimization problems to inscribe the largest m-ellipsoids that fit in each of the ${}^{d}C_{m}$ m-polytopes.
5. Re-associate the final axis lengths with singular vector directions. (appendix A2.4)

The ellipsoid fitting workflow in this thesis was developed as a combination of ideas from two publications:

1. Director & Hatchel (1977) [9] (sourced from Wai-Kai Chen (2009) [4]) led to the idea that the $2 \times d$ extremapoints obtained at the end of exploration could be treated as a d-polytope that constrains an inscribed d-ellipsoid.
2. Karl et al. (1994) [27] led to the idea that inscribing a single ellipsoid in a d-dimensional space (which as mentioned at the start of this chapter is computationally intractable) can be replaced with the problem of inscribing multiple lower dimensional ellipsoid cross-sections that can be reconstructed. Although to be clear, in the context of this thesis the term "reconstruction" simply means re-associating the axis lengths found through convex optimization (appendix A2.4) with the singular vector directions that are known before hand.

In the experiments, m is chosen as 2. This way, the m-dimensional ellipsoid becomes a 2-ellipsoid that is inscribed in a 2-polytope (or a regular 2D polygon). With the 40 dimension reservoir case, the optimization problem consists of inscribing ${}^{40}C_{2} = 780$ 2-ellipsoids in 2-polytopes. To serve as reference choosing $m = 3$ with this same example means the optimization problem consists of inscribing ${}^{40}C_{3} = 9880$ 3-ellipsoids in 3-polytopes. Increasing the value of m increases the number of constraints. The upper bound is $m = d$, in which case the single m-polytope has as many dimensions and facets as the initial d-polytope.

4.2 Validation

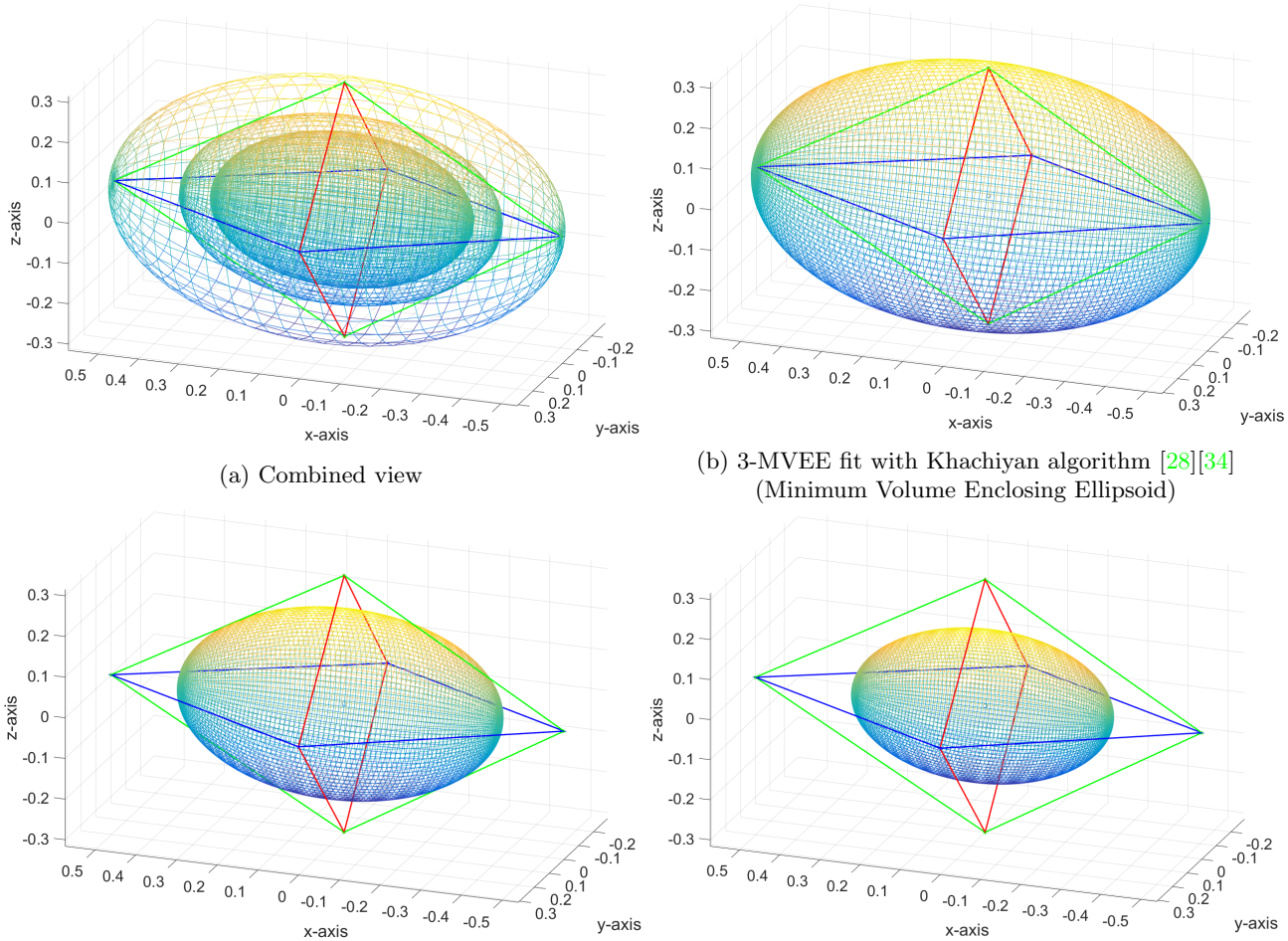
The purpose of validation is to test the quality of ellipsoid fit in the feasible region. The workflow and code presented in Dezert et al. [8] are used to generate a uniform distribution of samples in the interior of the ellipsoid. This workflow uses the symmetric positive definite ellipsoid matrix $\boldsymbol{Q}$ as the covariance matrix $\boldsymbol{\Sigma}$ for a multivariate distribution. As a result the shape (orientation/axes lengths) of the ellipsoid is taken into account in sample generation.

An alternative to sampling in the interior of the ellipsoid is sampling only on the ellipsoid surface. For this, the uniform distribution of points generated in the ellipsoid interior are projected to the ellipsoid surface.

(appendix A2.6) Such an approach has two benefits:

1. Sampling is now limited to the $d-1$ dimensional surface of the d-ellipsoid and the reduced dimensionality means more surface can potentially be covered with the same sampling budget.
2. The distance of the samples generated relative to the optimum coordinate at the ellipsoid centre is maximized.

Results of testing the two sample generation approaches on the Rosenbrock function have been included in Appendix A2.6. In the context of this thesis, the fit quality is just a ratio that compares the number of feasible samples to the total number of samples. Each sample counts as one function evaluation and represents a unique control vector coordinate in d-dimensional space.



(a) Combined view

(b) 3-MVEE fit with Khachiyan algorithm [28][34] (Minimum Volume Enclosing Ellipsoid)

(c) 3-CSC-MVIE

(d) 3-MVIE

Figure 4.6: Comparison of ellipsoid sizes - As seen, MVEE > CSC-MVIE > MVIE

Chapter 5

Results

5.1 Optimization

Optimization is performed in the log transformed space. The EnOpt gradient is used along with strong Wolfe line search and the BFGS scheme approximates the Hessian. The production period is 4000 days divided in 10 time steps of 400 days each. Using equation 2.3 this results in an optimization problem consisting of 40 variables. ($d = 4$ wells $\times$ 1 ICV/well $\times$ 10 time steps $= 40$)

An ensemble of 12 control vectors is used. All controls are set at 0.5 (half open state) at the initial iteration. This initial setting is converted to the log transformed space using equation 3.3. The optimizer is allowed to remain at a coordinate for **three** successive iterations without an increase in function value. After this a new coordinate is accepted even if it results in a decrease in NPV.

Refer to section 2.6 for the initial reservoir parameters.

c_{slo}	0.1
c_{slc}	1
Wolfe c_1	10^{-4}
Wolfe c_2	0.9
Wolfe iterations	5
Zoom iterations	8
Initial Hessian	Identity

Table 5.1: Optimizer configuration

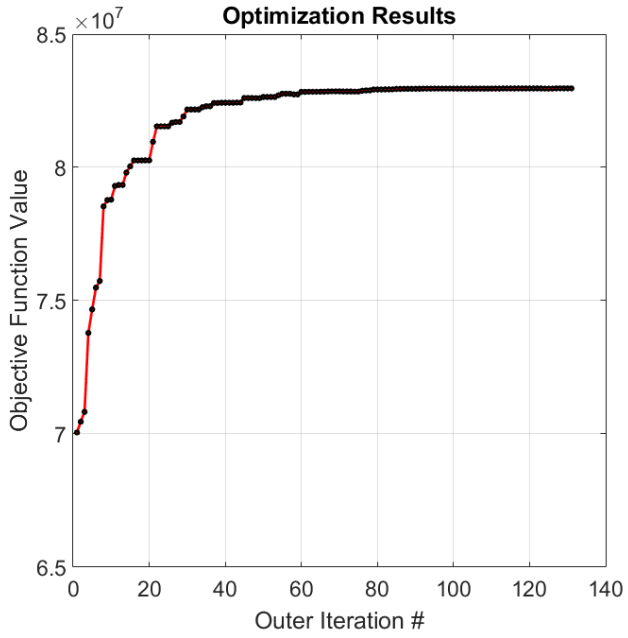
u^{min}	10^{-6}
u^{max}	1
$u^{min,\text{perturb}}$	10^{-4}
$u^{max,\text{perturb}}$	0.9999

Table 5.2: Bound Specification

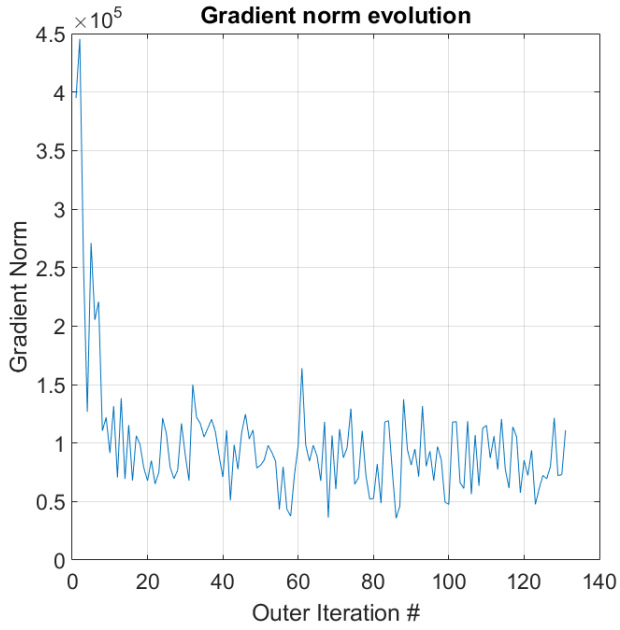
c_{slo} and c_{slc} are used to calculate the trial step length (appendix A1.2) for strong Wolfe line search. $u^{min,\text{perturb}}$ and $u^{max,\text{perturb}}$ are heuristically determined parameters (appendix A1.4) which limit the minimum and maximum magnitudes of controls in the perturbed control vector ensemble.

Optimization iterations	131
Final NPV	$\$8.2965 \times 10^7$
Final gradient norm	1.1091×10^5
BFGS updates	62

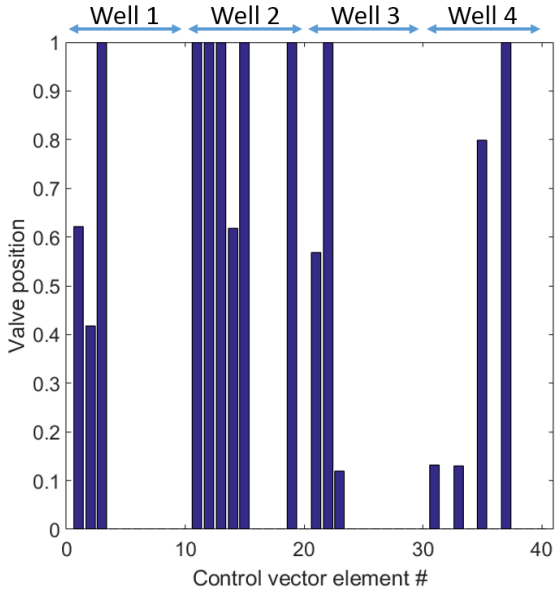
Table 5.3: Optimization results



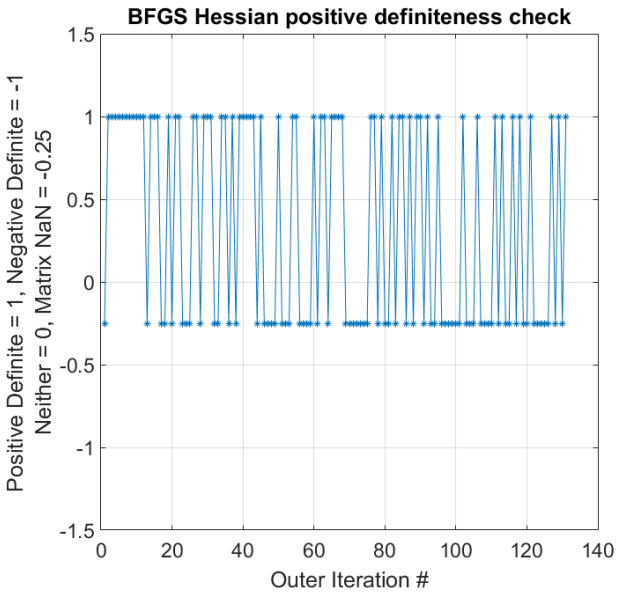
(a) Function value evolution



(b) EnOpt gradient norm evolution



(c) ICV settings @ iteration 131

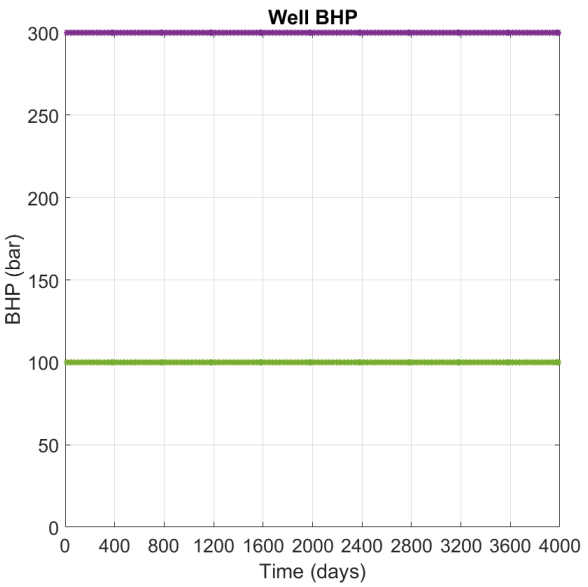


(d) BFGS positive definiteness check

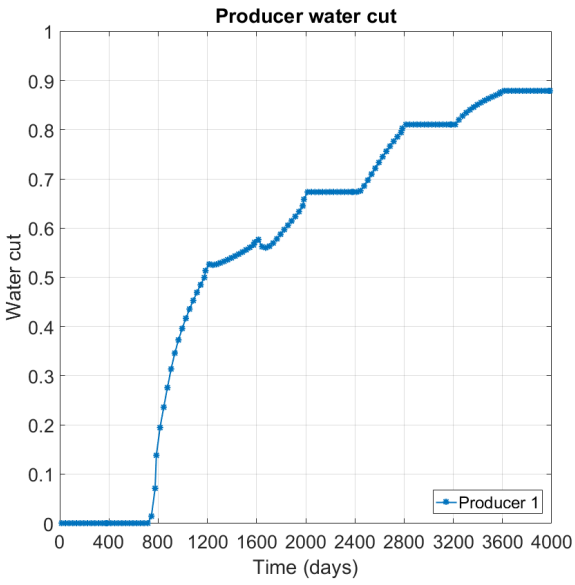
Figure 5.1: Optimization results

Figures 5.1a and 5.1b present the function value and gradient norm evolution over optimization iterations. A total of 62 BFGS updates occur which satisfy both the sufficient increase and curvature conditions. As the number of valid updates is greater than the number of controls ($d = 40$), the criteria suggested in Gill et Al. (2001) [18] - relating to the minimum number of iterations required for the BFGS Hessian to fully capture curvature information - is met. Optimization was stopped at iteration 131 since no further increase in NPV was observed.

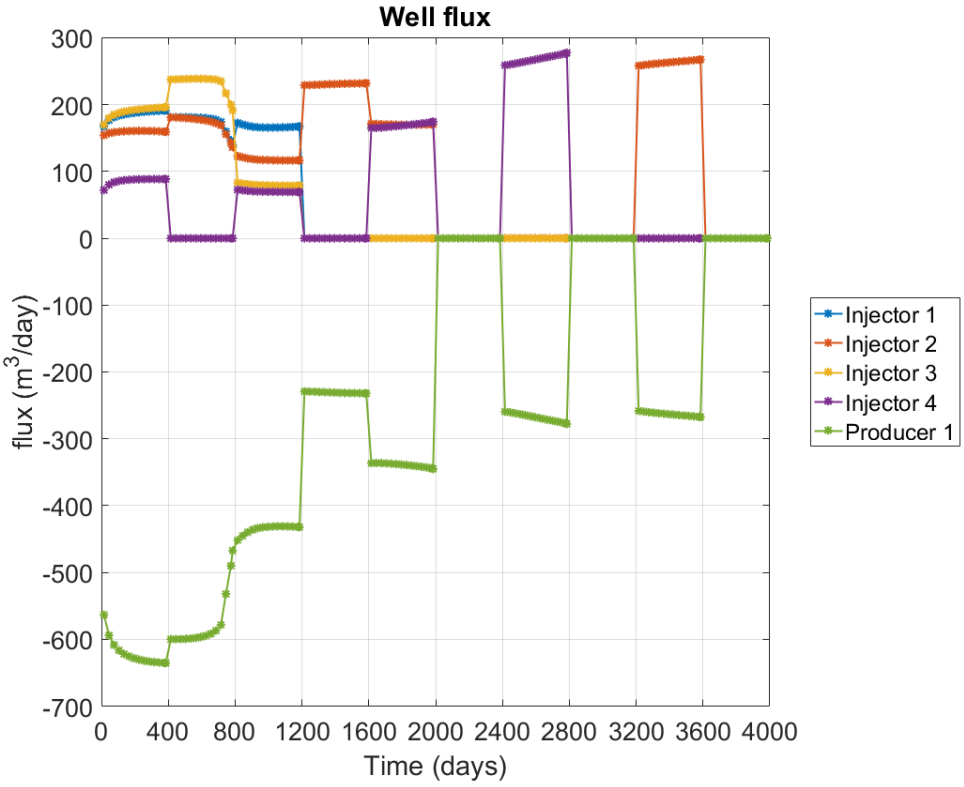
Figure 5.1c indicates the valve settings in each well at each time step. With 10 timesteps, 10 controls per well are obtained. The eigen values of each BFGS Hessian are computed to verify if the matrix is indeed positive definite. A successful test results in a value of '1' in figure 5.1d. In case a BFGS update does not occur due to either Wolfe condition not being met, the BFGS Hessian for that iteration is declared `NaN`. This corresponds to a value of '-0.25' in figure 5.1d. Values of '0' and '-1' respectively represent mixed and negative definite matrices - neither of which are encountered.



(a) Well BHP



(b) Water cut

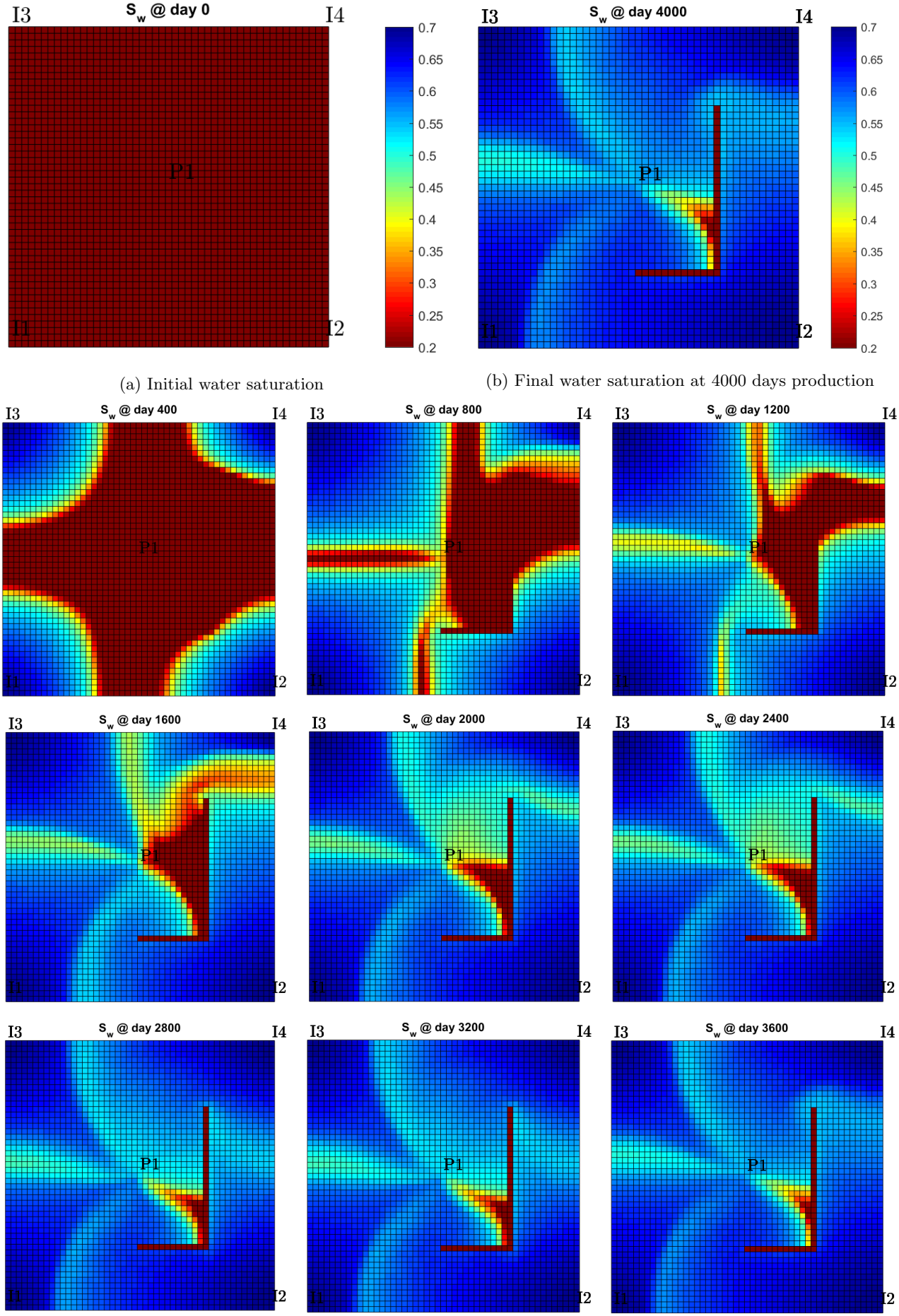


(c) Well flux

Figure 5.2: Production results corresponding to ICV controls at optimization iteration 131

As mentioned in section 2.6, figure 5.2a indicates that injectors are fixed at 300 bars BHP while the producer operates at 100 bars BHP. The control time step is 400 days while the simulator time step is a maximum of 30 days. Figure 5.2b shows water production beginning around 730 days and water cut at the end of production being 87.93%.

Figure 5.2c indicates that the producer is shut in (zero flux) over 3 intervals: 2000 to 2400 days, 2800 to 3200 days and 3600 to 4000 days. This result is most likely explained by the fact that both water injection and water disposal are charged, meaning it is efficient from the perspective of maximizing NPV to simply shut in the well.

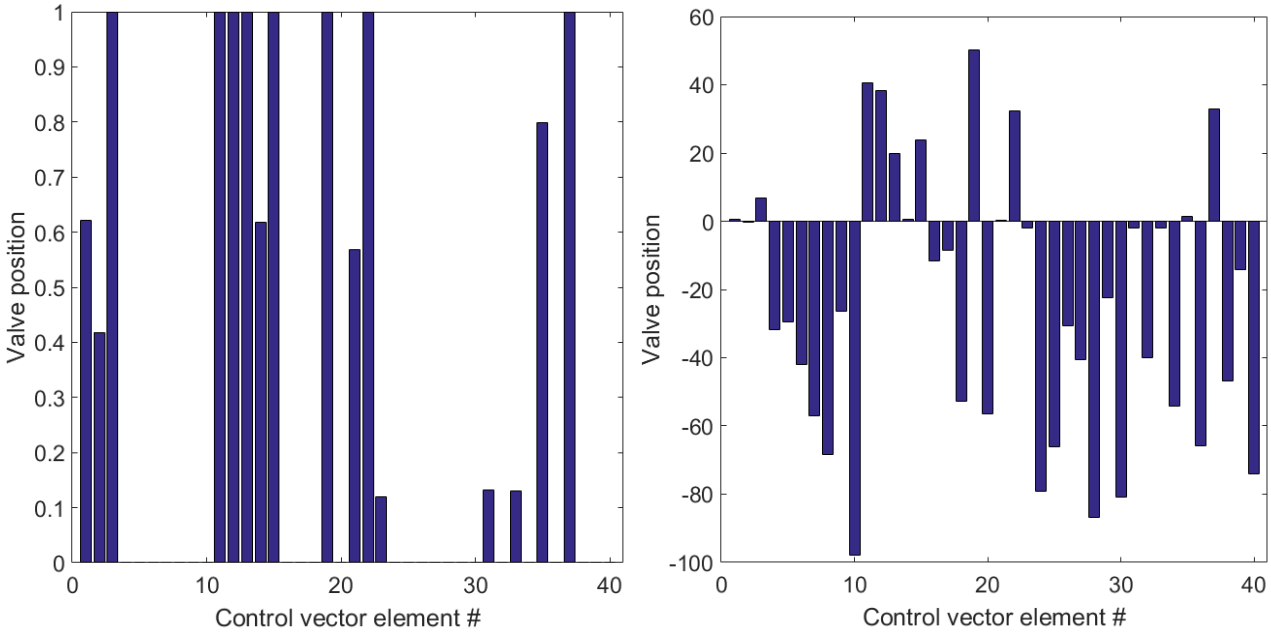


(a) Initial water saturation

(b) Final water saturation at 4000 days production

Figure 5.3: Reservoir water saturation $S_{wc} = 0.2$, $S_{w,\max} = 1 - S_{or} = 1 - 0.3 = 0.7$. This information is used to set the color axis scale as [0.2 0.7] The intermediate saturation snapshots are also presented.

5.2 Exploration



(a) Regular control space

(b) Log transformed space

Figure 5.4: Optimum control vector in regular and log transformed space

The SVD of the BFGS Hessian at the optimum provides the search directions for exploration in log transformed space. The log transformed control vector in figure 5.4b serves as the exploration origin. Three unique runs of exploration are performed where up to 0.5%, 1.0% and 2.0% decrease in NPV relative to the optimum are accepted.

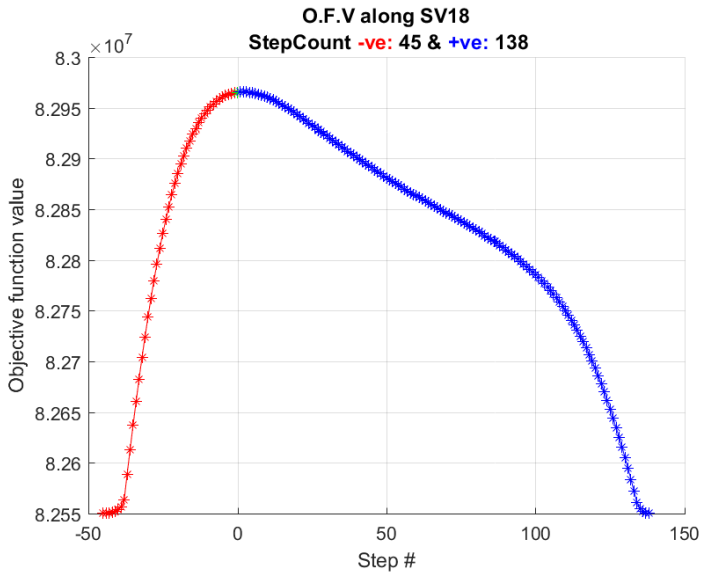
Optimum NPV	$\$8.2965 \times 10^7$
Run 1 $J_{\min}^{0.5\%}$ (0.5% decrease)	$\$8.2550 \times 10^7$
Run 2 $J_{\min}^{1.0\%}$ (1.0% decrease)	$\$8.2136 \times 10^7$
Run 3 $J_{\min}^{2.0\%}$ (2.0% decrease)	$\$8.1306 \times 10^7$

Table 5.4: Exploration criteria

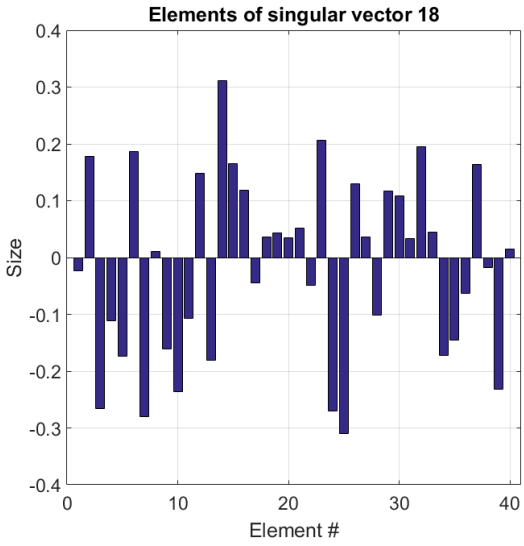
% decrement	Step size	Function evaluations
0.5%	$\alpha_{\text{init}}^{0.5\%} = 0.25$	5524
1.0%	$\alpha_{\text{init}}^{1.0\%} = 1.0$	2815
2.0%	$\alpha_{\text{init}}^{2.0\%} = 2.0$	2886

Table 5.5: Function evaluations performed for exploration along all d singular vectors - Minimum accepted step size is $\alpha_{\min} = 10^{-4}$

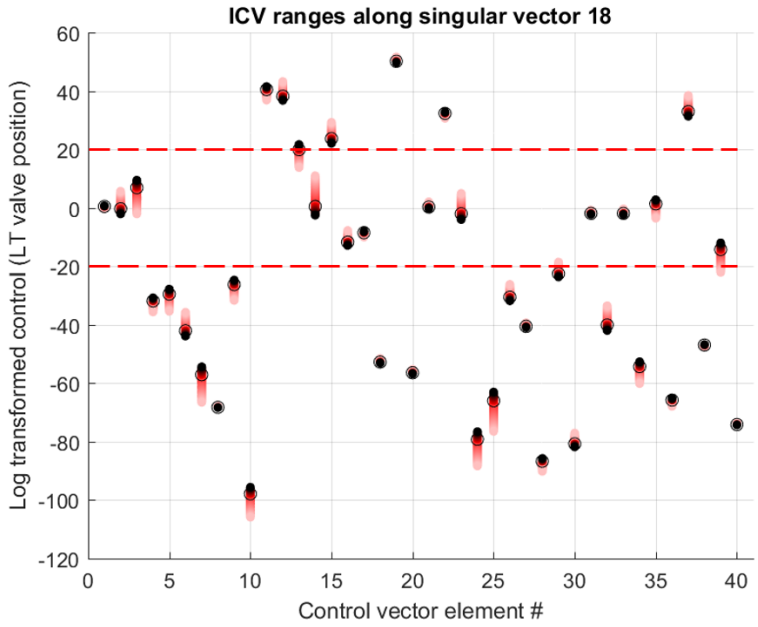
To gain intuition of exploration in a multivariate sense, the change in controls and associated objective function values along an exploration direction can be visualized. This information is presented in figures 5.5-5.8. While exploration is performed along all 40 singular vectors, only the results corresponding to singular vector 18 (which is arbitrarily selected) are displayed.



(a) Objective function value (O.F.V) evolution with exploration (step size = 0.25)



(b) Elements of singular vector 18

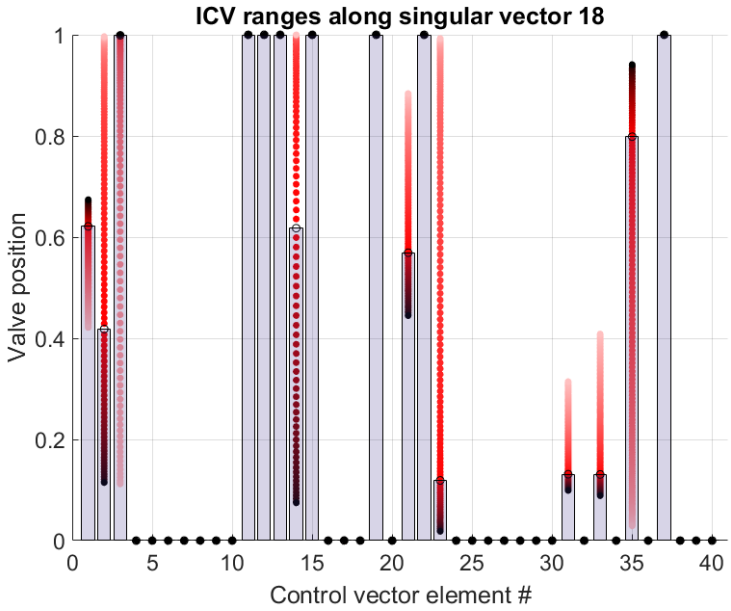


(c) Exploration ranges in log transformed space

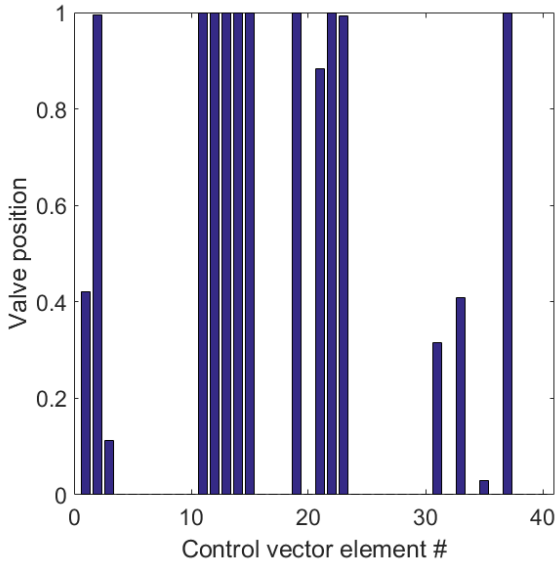
Figure 5.5: (Part 1) Exploration till 0.5% decrement ($\$8.2550 \times 10^7$) along singular vector 18

The algorithm presented in equation 3.2 is used for exploration. Each exploration step corresponds to a unique d-dimensional control vector that differs slightly with respect to the optimal controls. Figure 5.5a indicates the objective function values associated with each such exploration step - along singular vector 18 (SV18). Figure 5.5a can also be interpreted as the visualization of a 1D line search procedure where SV18 is the search direction. The objective function values for exploration steps along the positive and negative extents of SV18 are indicated in blue and red respectively. The effect of step size contraction near the feasible region boundary is seen with the relatively large number of steps concentrated around the $\$8.2550 \times 10^7$ mark.

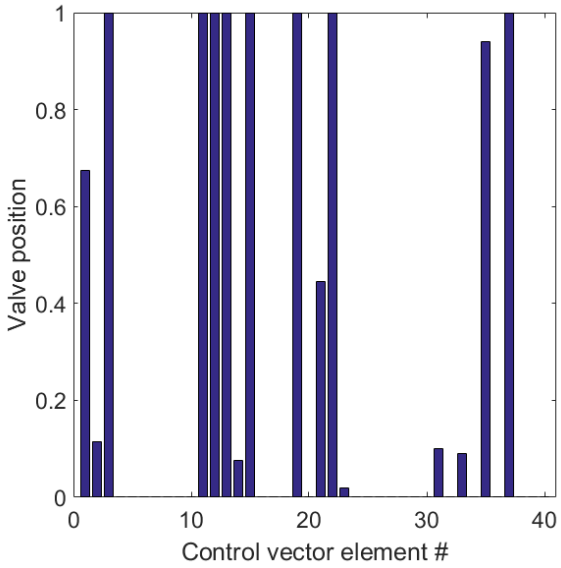
Figure 5.5b presents the elements of singular vector 18. The Euclidean norm of this vector (as with all singular vectors) is 1. For a given step size, the sign and magnitude of a singular vector element determine the change in value of the corresponding (log transformed) control vector element. Figures 5.5c and 5.6a present respectively the (same) change in controls along SV18 in two different spaces - the log transformed space and regular control space. At each control, exploration steps along the positive extent are indicated by progressively lighter red while exploration steps along the negative extent are indicated by progressively darker red. Common shades of red across all 40 controls therefore indicate one exploration step/control vector.



(a) Exploration ranges in regular control space with optimum control vector from figure 5.4a overlayed



(b) Controls at final positive step (lightest red in figure 5.6a)



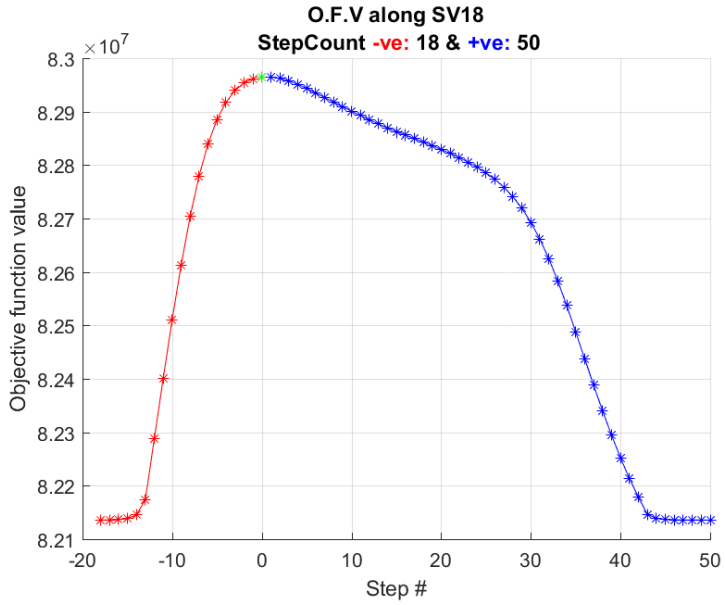
(c) Controls at final negative step (darkest red in figure 5.6a)

Figure 5.6: (Part 2) Exploration till 0.5% decrement ($\$8.2550 \times 10^7$) along singular vector 18

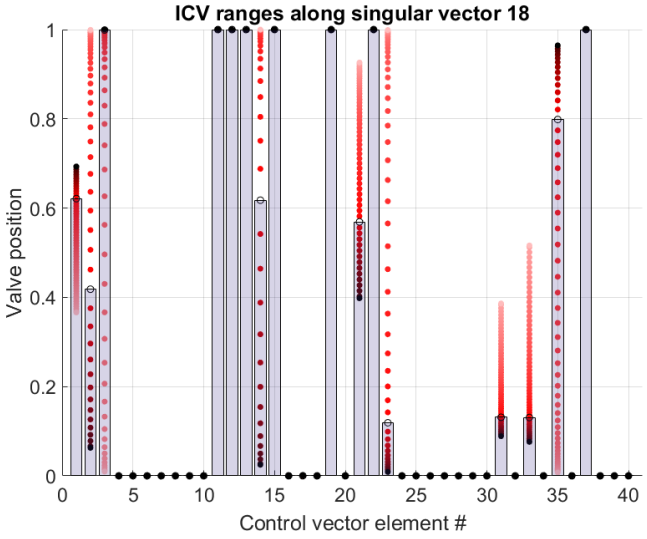
From figures 5.5 and 5.6, a few observations can be made:

1. As exploration is performed in the log transformed space, the change in controls at each exploration step when visualized in the regular control space is non-linear.
2. Controls with a large magnitude in the log transformed space remain unchanged in the regular space. Only controls with magnitudes within a narrow range in the log transformed space (indicated by red dashed lines in figure 5.5c) exhibit change in the regular space during exploration.
3. Exploration along the positive and negative extents of one singular vector results in two (extrema) points on the boundary of the feasible region. The two extremapoints are connected by a continuous set of feasible control vectors and for the step size selected ($\alpha_{\text{init}} = 0.25$), the objective function value changes smoothly in between.
4. The control vectors associated with the positive (figure 5.6b) and negative (figure 5.6c) extremapoints have identical objective function values - although they represent fairly different control strategies.

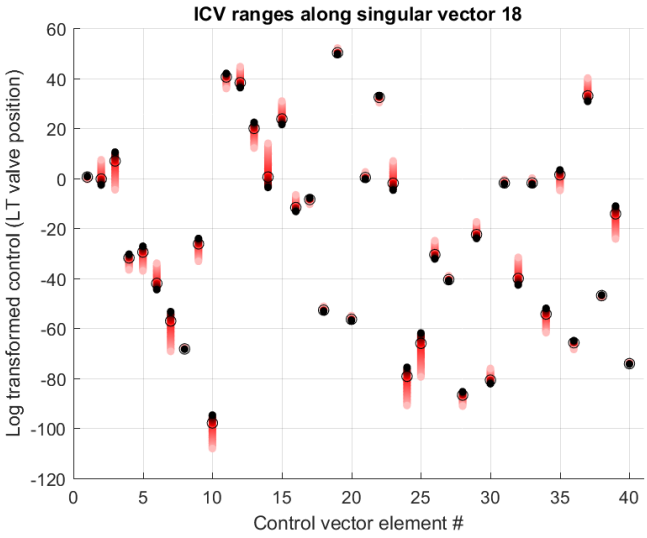
These observations both validate the idea that lifecycle optimization is ill-posed and demonstrate that feasible control strategies can be identified by exploring the Hessian null space. Figures 5.7 and 5.8 present similar exploration results where a 1.0% and 2.0% decrement is accepted relative to the optimum.



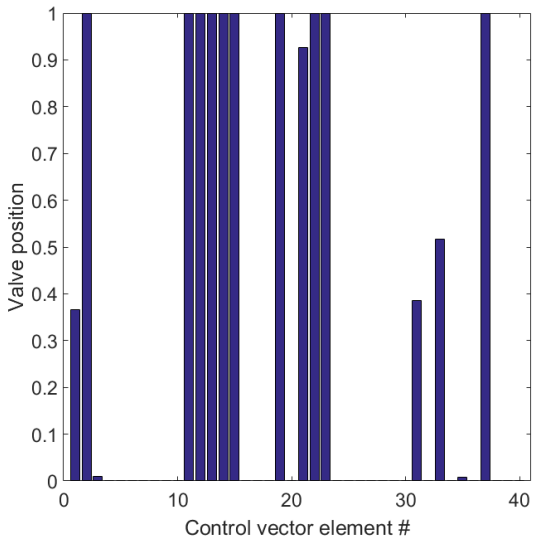
(a) Objective function value (O.F.V) evolution with exploration (step size = 1)



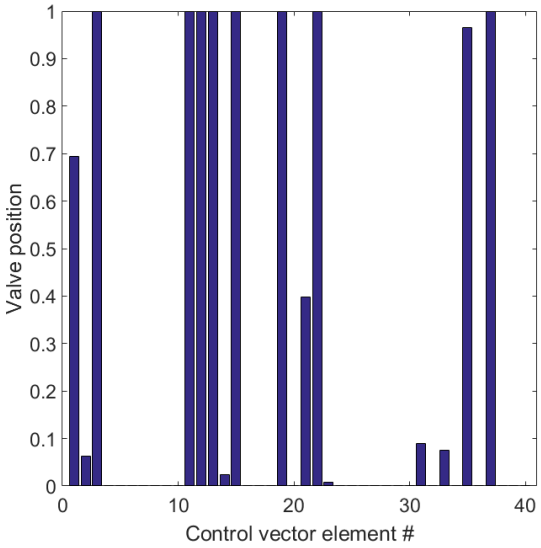
(b) Exploration ranges in regular control space



(c) Exploration ranges in log transformed space

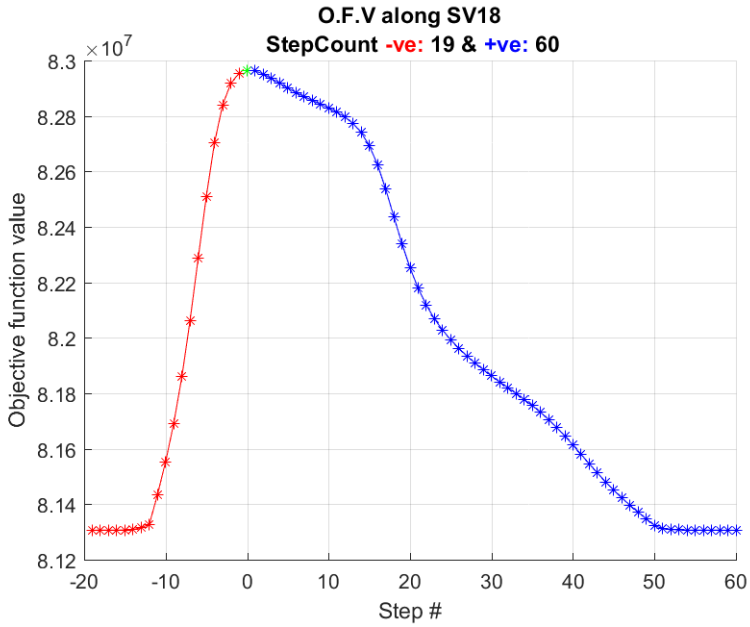


(d) Controls at final positive step (lightest red in figure 5.7b)

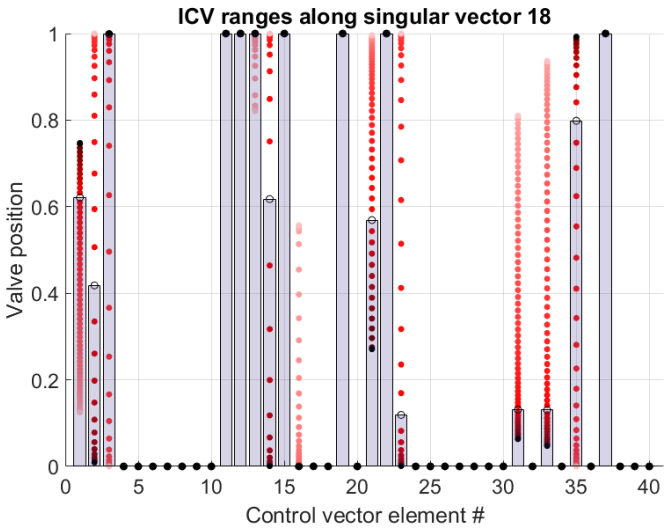


(e) Controls at final negative step (darkest red in figure 5.7b)

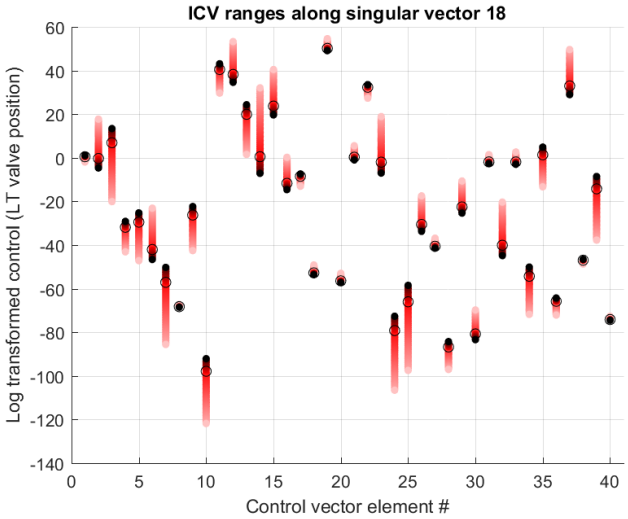
Figure 5.7: Exploration till 1.0% ($8.2136 × 10^7) decrement along singular vector 18



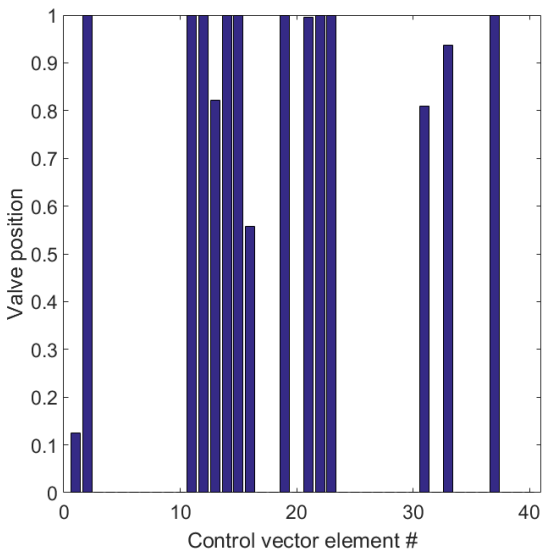
(a) Objective function value (O.F.V) evolution with exploration (step size = 2)



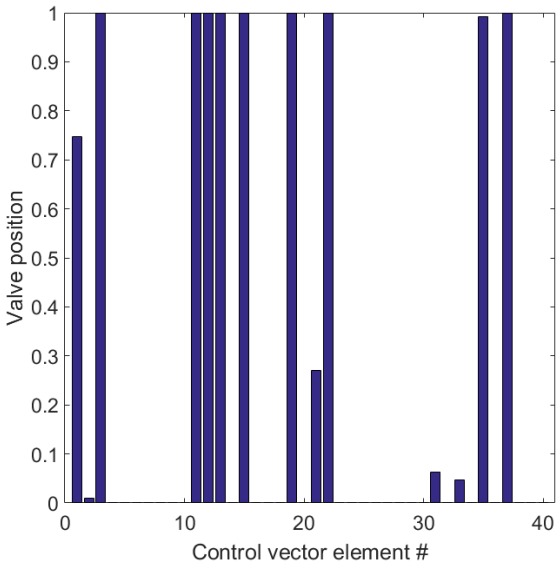
(b) Exploration ranges in regular control space



(c) Exploration ranges in log transformed space



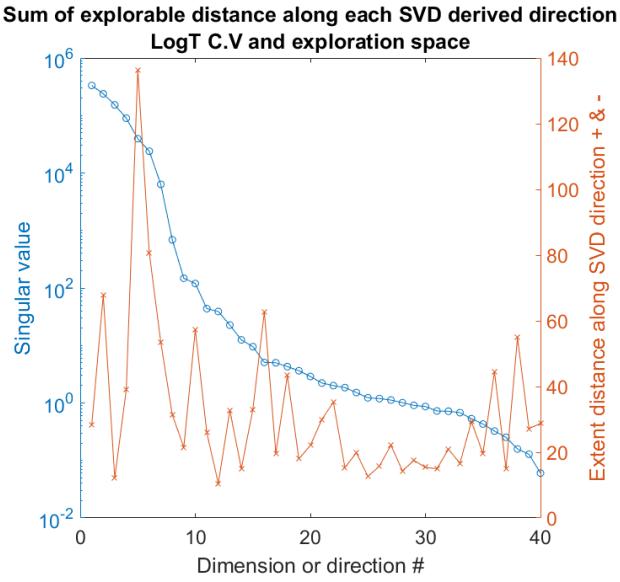
(d) Controls at final positive step (lightest red in figure 5.8b)



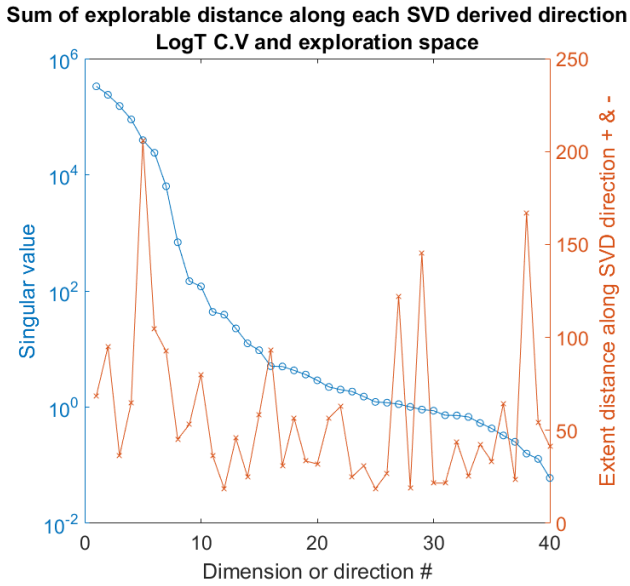
(e) Controls at final negative step (darkest red in figure 5.8b)

Figure 5.8: Exploration till 2.0% decrement ($\$8.1306 \times 10^7$) along singular vector 18

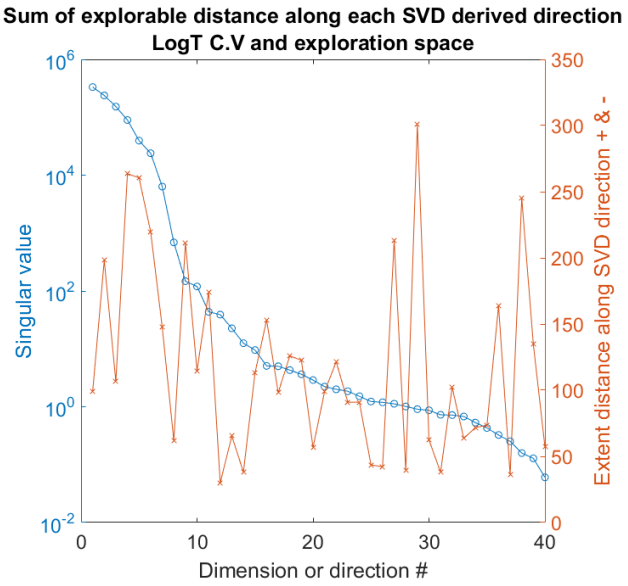
5.2.1 Explorable distances - Reservoir case



(a) Exploration till 0.5% decrement ($8.2550 $\times 10^7$)



(b) Exploration till 1.0% decrement ($8.2136 $\times 10^7$)



(c) Exploration till 2.0% decrement ($8.1306 $\times 10^7$)

Figure 5.9: Singular value magnitude vs. total explorable distance - Reservoir case. LogT C.V stands for log transformed control vector and indicates that exploration is performed in the log transformed space

Figure 5.9 compares the singular value magnitudes of the BFGS Hessian against the total explorable distance along each singular vector. The singular value magnitudes are listed on the left log Y-axis. The explorable distance for a given singular vector is measured as the euclidean distance between its two extremapoints in the log transformed space. This is listed on the right Y-axis. (Extremapoints are the final feasible coordinates on positive and negative exploration extents of a singular vector. Exploration along one singular vector produces two extremapoints on the boundary of the feasible region) Figure 5.9 leads to a few observations:

1. Larger accepted decrements from the optimum NPV result in longer exploration distances.
2. The singular value magnitudes are identical across figures 5.9a-5.9c since the same BFGS Hessian is used for all three exploration tests.
3. Unlike in the 40D Rosenbrock case (appendix A1.3.5) the singular value magnitudes do not correlate with explorable distances - as in, a smaller singular value does not translate to a larger explorable distance along the associated singular vector direction.

While it is possible that the stochastic nature of the EnOpt gradient results in a poor BFGS Hessian approximation (thereby resulting in the lack of correlation between singular value magnitudes and explorable distances), it is more likely that the Hessian SVD derived curvature/null space information is only "locally" valid. In this context, "locally" refers to small perturbations in the immediate vicinity of the optimal control vector where the Taylor series derived quadratic objective function approximation stated in equation 2.18 holds valid.

Exploration on the contrary involves large steps away from the optimum and also takes into account the shape/size (asymmetry and non-convexity - See in 5.3) of the feasible region since the extremapoints are always sampled on the feasible region boundary. It should also be noted that the shape/size of this feasible region depends on a user specified function value threshold (See equation 3.1) and that the Hessian can in no way capture this information.

5.3 2D plane scan and ellipsoid fitting

The $2 \times d$ extremapoints from each of the three exploration runs are independently used as inputs to the ellipsoid fitting workflow described in chapter 4. With $d = 40$ controls in the reservoir case and constraint polytope dimension $m = 2$, the convex optimization problem consists of inscribing ${}^{d}C_{m} = {}^{40}C_{2} = 780$ 2-ellipsoids in 2-polytopes - for each exploration run. Similarly, ${}^{40}C_{2}$ unique 2D plane scans can be performed with the algorithm described in 3.5 and 3.6.

For a given pair of singular vectors, the constraint 2-polytope and inscribed 2-ellipsoid generated by the ellipsoid fitting workflow is overlayed on the corresponding 2D plane scan - which now serves as a (visual) diagnostic to identify the quality of fit for the inscribed 2-ellipsoid in the feasible region of that 2D plane. Since the number of function evaluations required to generate ${}^{40}C_{2}$ 2D plane scans is computationally intractable, only the results for a few arbitrarily selected singular vector combinations are presented.

Plane	0.5%	1.0%	2.0%	Total Samples (Decrement %)
P1S11	116	501	1829	1991 (2.1%)
P5S28	273	563	1758	1882 (2.1%)
P4S2	381	1004	6325	28734 (3.0%)

Table 5.6: Number of feasible samples in each 2D plane scan result. 'P' and 'S' indicate singular vectors corresponding to primary and secondary exploration directions. Plane scan step size is set as $\alpha_{\text{init}} = 2.5$

Each 2D plane scan is performed till a maximum decrement of 2.1% ($\$8.1223 \times 10^7$ - P1S11,P5S28) or 3.0% ($\$8.0476 \times 10^7$ - P4S2) relative to the optimum NPV ($\$8.2965 \times 10^7$). Using this data, the 2D plane scans for 0.5%, 1.0% and 2.0% decrements are generated as the appropriate subset of feasible samples. The "extra" data is used mainly to improve the distinction between feasible/non-feasible regions.

In the following pages, figures 5.10-5.12 elaborate the relation between 1D exploration and 2D plane scans. Figures 5.13, 5.14-5.15 and 5.16-5.17 present respectively the plane scan results for P1S11, P4S2 and P5S28. The 2-polytope and 2-ellipsoid generated by the ellipsoid fitting workflow for each 1D exploration result (0.5%, 1.0%, 2.0%) are overlayed on the corresponding 2D plane scan. The scaling constant for ellipsoid fit specified in algorithm A2.19 is set as zero to ensure the largest inscribed 2-ellipsoid. (Refer to figure 5.19b and A2.4 to see the effect of a non-zero scaling constant) The centre of each 2-ellipsoid is also fixed as the exploration origin/optimum and is indicated by a green dot.

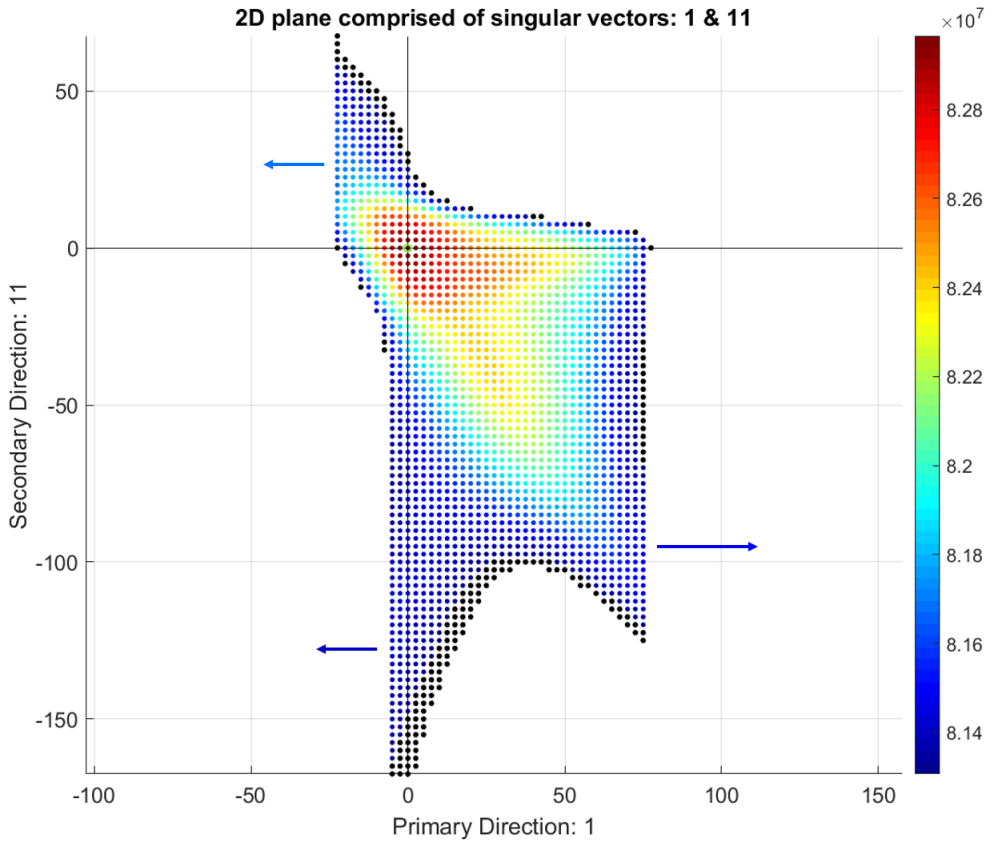
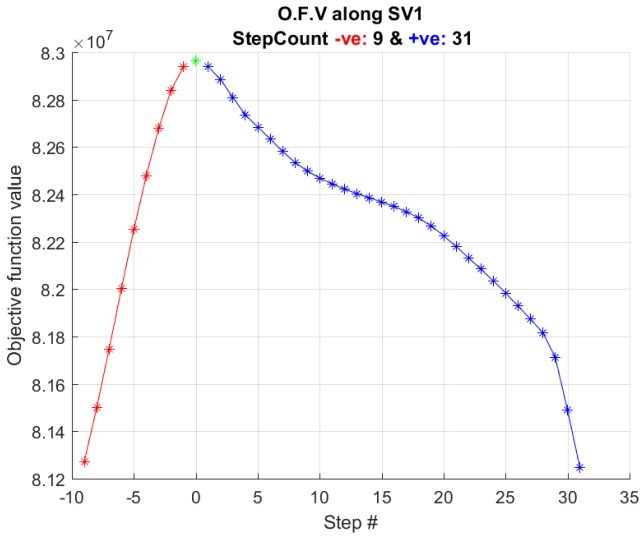
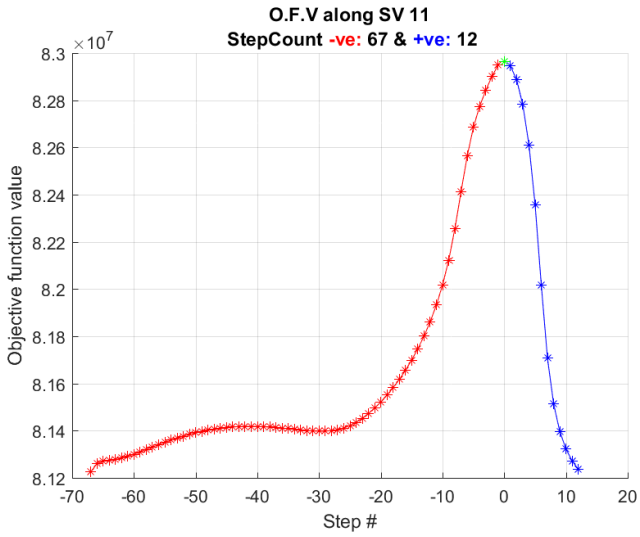


Figure 5.10: Exploration (plane scan) till 2.1% decrement ($\$8.1223 \times 10^7$). Samples with function value within 2.0% ($\$8.1306 \times 10^7$) of optimum NPV are indicated in color. The remaining samples are colored black to better distinguish the feasible region. The optimum is indicated by a green dot at the intersection of the two bold lines



(a) Function value change along SV1 (step size = 2.5) Exploration till 2.1% decrement ($\$8.1223 \times 10^7$)

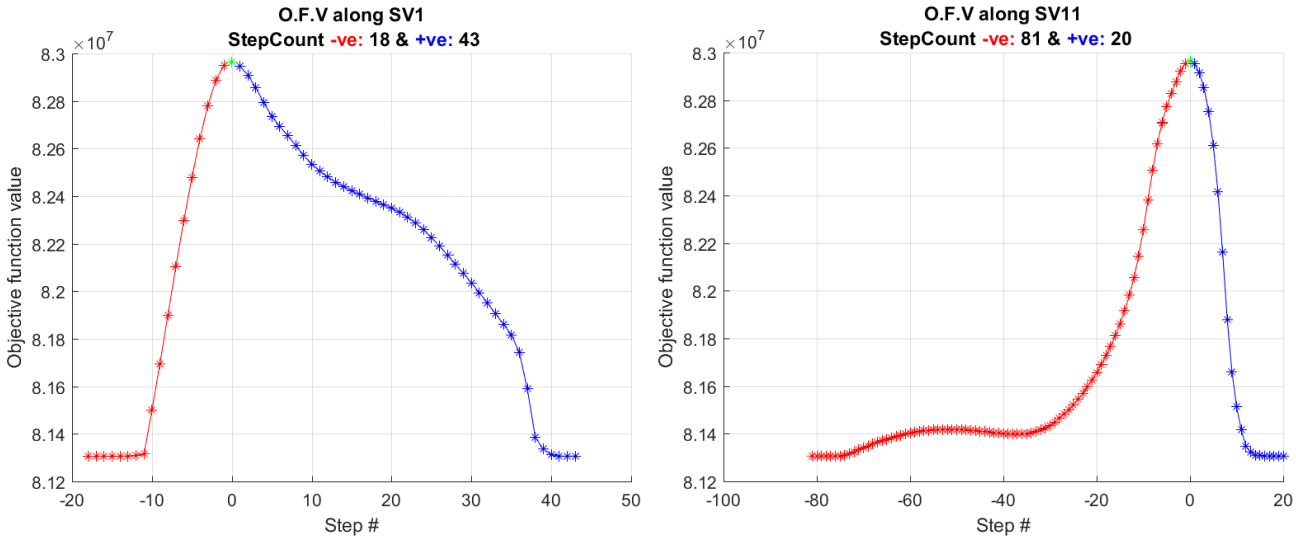


(b) Function value change along SV11 (step size = 2.5) Exploration till 2.1% decrement ($\$8.1223 \times 10^7$)

Figure 5.11: Function value change along axes of figure 5.10 - Horizontal axis in 5.11a & Vertical axis in 5.11b

In figure 5.10, singular vector 1 is used to perform primary exploration and the feasible samples found are indicated along the horizontal line (X-axis). All primary exploration samples are then used as initial coordinates for secondary exploration along positive and negative extents of singular vector 11 - which is indicated vertically. As mentioned in section 3.3, figure 5.10 is a proxy 2D plot since exploration in d-dimensional space cannot be visualized directly. This means, each sample in 2D is associated with a d-dimensional control vector. If (X, Y) is the coordinate of a sample on the 2D plane, the associated d-dimensional control vector is located at a distance of X units along singular vector 1 and Y units along singular vector 11 away from the optimum control vector.

The process of representing d-dimensional coordinates in a lower m-dimensional space where $m = 2$ is identical to the approach followed by the ellipsoid fitting workflow (See chapter 4 and appendix A2.1). This makes it possible to overlay the 2-polytope and inscribed 2-ellipsoid generated by the ellipsoid fitting workflow on the corresponding 2D plane scan for a given pair of singular vectors.



(a) Function value change along SV1 (step size = 2) Exploration till 2.0% decrement ($8.1306 × 10^7)

(b) Function value change along SV11 (step size = 2) Exploration till 2.0% decrement ($8.1306 × 10^7)

Figure 5.12: Objective function value (O.F.V) evolution in 1D exploration

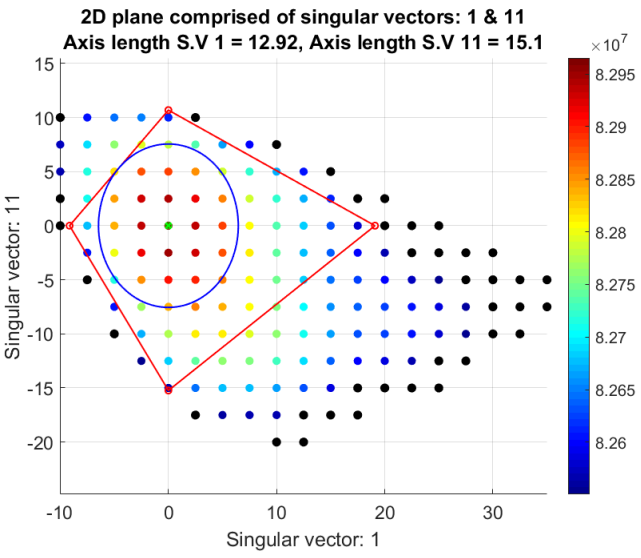
The bold horizontal/vertical lines in figure 5.10 highlight function value change for exploration steps along the singular vector axes (in terms of marker color). For better readability, this result is reproduced in figures 5.11a and 5.11b. As expected, the function value change along each (singular vector) axis of the 2D plane scan is nearly identical to the corresponding 1D exploration result in figures 5.12a (SV1) and 5.12b (SV11). Minor differences arise from the fact that the step sizes used are different: $\alpha = 2$ in 1D exploration and $\alpha = 2.5$ in 2D plane scan. Additionally, the 2D plane scan does not use step size contraction and exploration is performed till a 2.1% ($8.1223 × 10^7) decrement relative to the optimum.

The arrows in figure 5.10 indicate that for the selected decrement in function value relative to the optimum (2.1%), the non-convex feasible region is not fully explored. This means if the primary and secondary exploration directions are switched, the overall shape of the 2D plane scan will change. This issue is avoided in case of results with 0.5% and 1.0% decrements since exploration is performed till a relatively larger initial decrement ($\geqslant$ 2.1%) which therefore makes it possible to display the entire feasible sample subset.

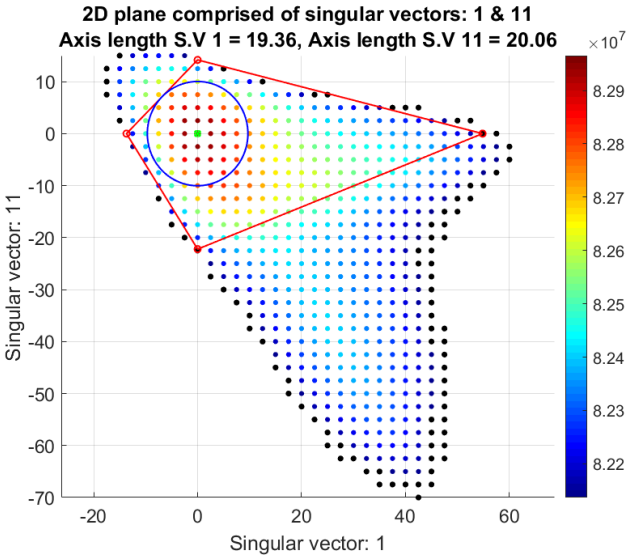
Two issues limit the usability of 2D plane scans in exploration:

1. The ellipsoid fitting workflow requires that each m-polytope is convex. With 1D exploration where two singular vectors produce 4 extremapoints per 2D plane - this requirement is easily met. In comparison, the 2D plane scan produces a much larger number of samples on the feasible region boundary - which as seen in figure 5.10 is non-convex. While the 'concave hull' algorithm presented in Galton & Duckham (2006) [16], Moreira & Santos (2007) [33] can be used to identify and extract the set of samples on the feasible region boundary, creating a convex approximation of this non-convex set still needs to be addressed.

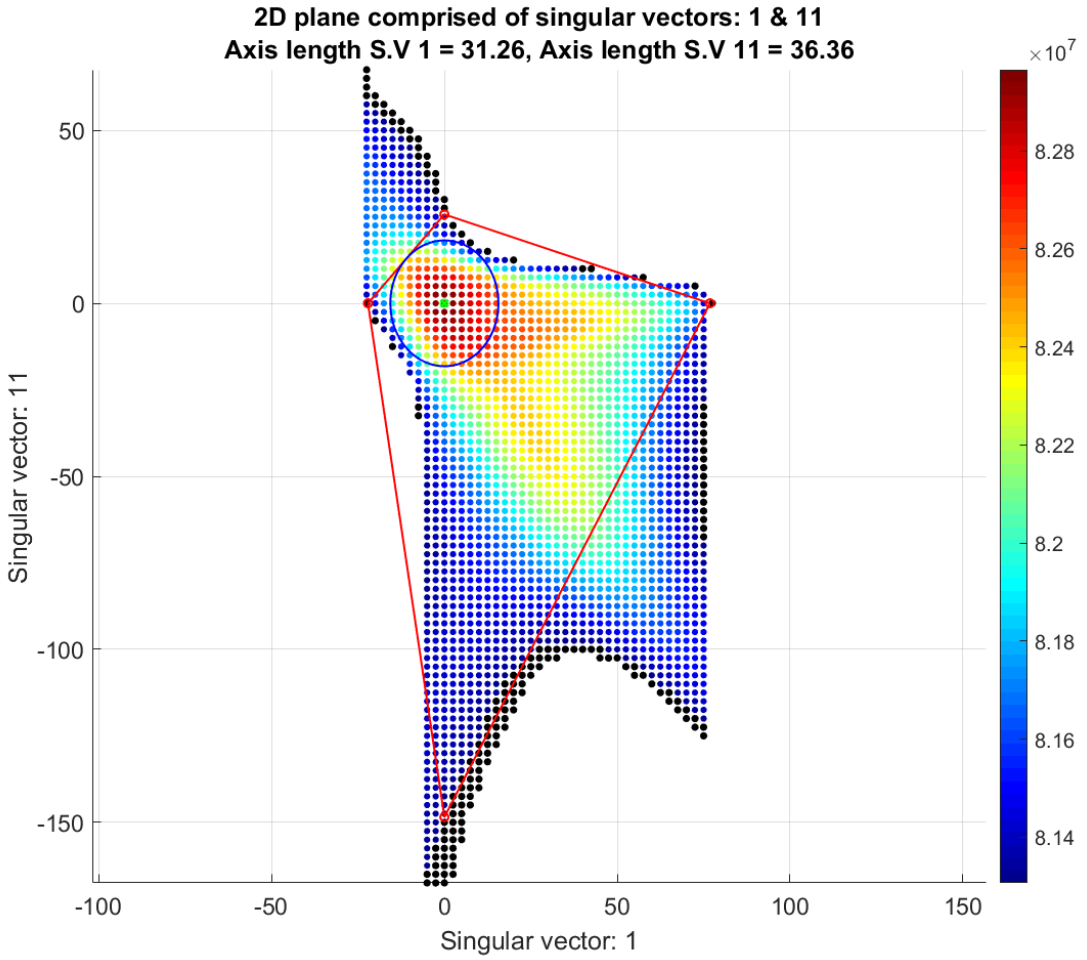
2. Computational cost: Tables 5.5 and 5.6 present respectively, the function evaluations used in each 1D exploration run and 2D plane scan. It is noted that the number of evaluations required to explore/map one 2D plane is sometimes greater than the number of evaluations used in a complete 1D exploration run. While larger step sizes and more efficient exploration algorithms are possible, ${}^{40}C_2 = 780$ unique 2D plane scans will still be required for the reservoir case with $d = 40$ controls.



(a) 0.5% accepted decrement ($8.2550 × 10^7$)
0.6% maximum decrement ($8.2467 × 10^7$)



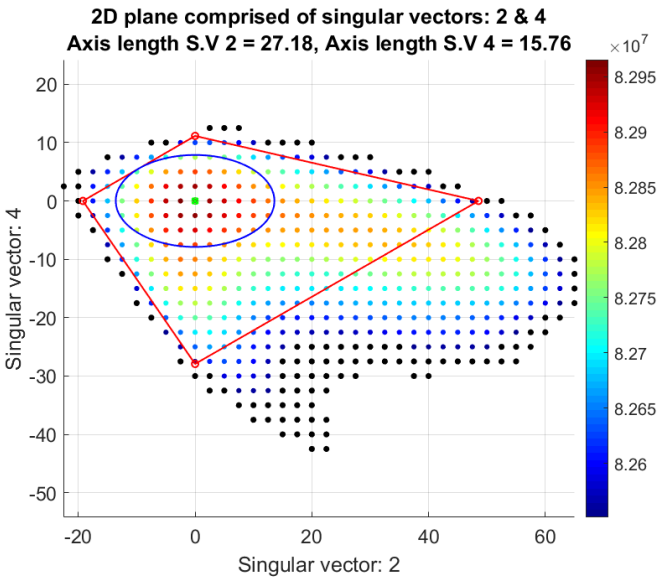
(b) 1.0% accepted decrement ($8.2136 × 10^7$)
1.1% maximum decrement ($8.2052 × 10^7$)



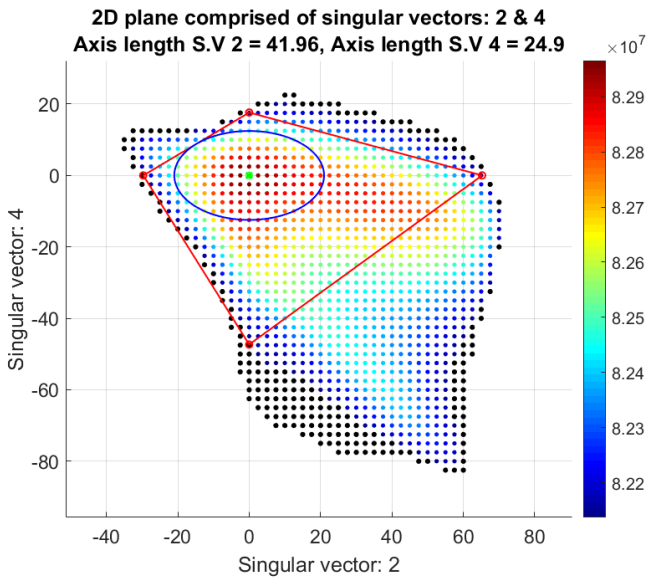
(c) 2.0% accepted decrement ($8.1306 × 10^7$)
2.1% maximum decrement ($8.1223 × 10^7$)

Figure 5.13: 2D plane scan result - Primary direction: SV1, Secondary direction: SV11

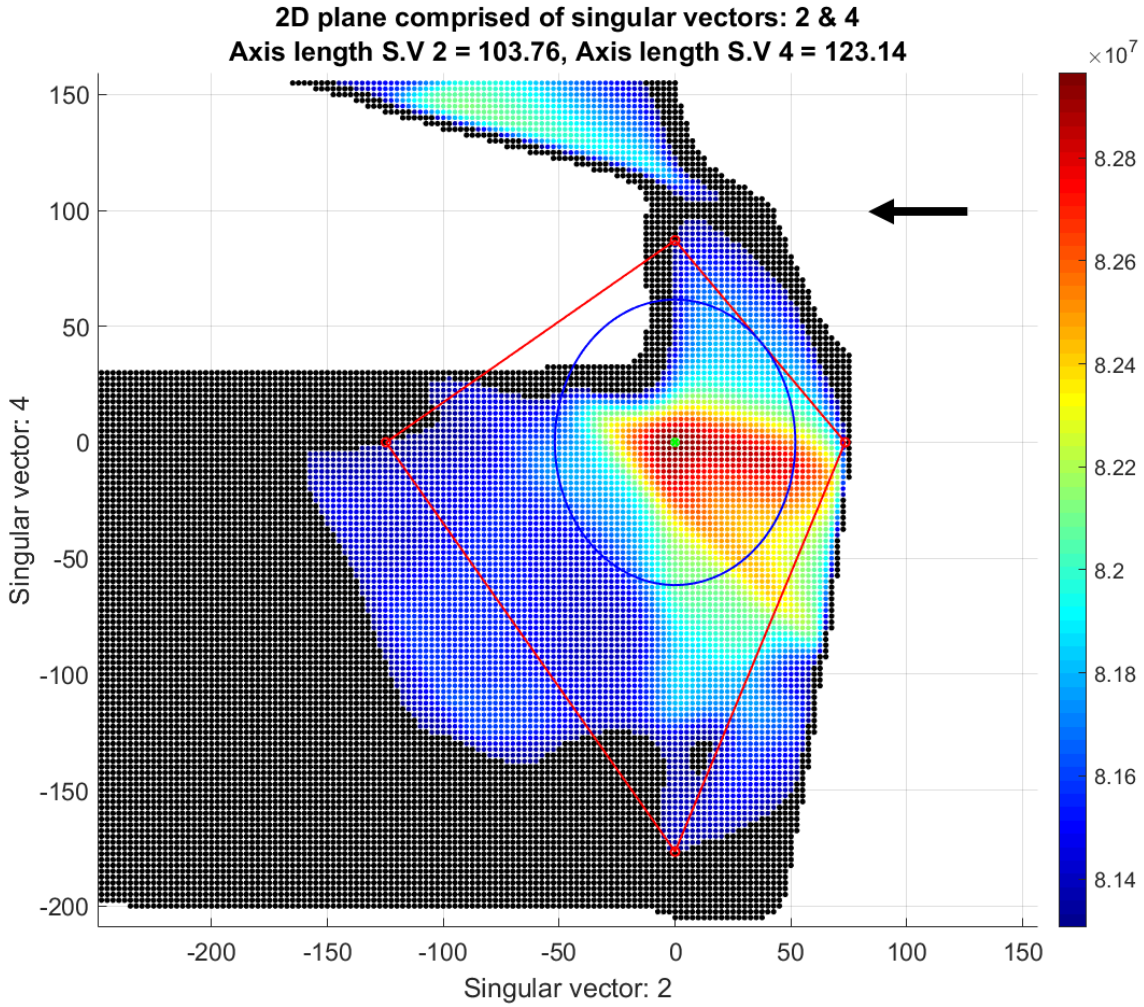
Samples within the accepted decrement threshold (0.5%,1.0% or 2.0%) are indicated in color. Samples with objective function value smaller than this threshold but still larger than the maximum decrement threshold (0.6%,1.1% or 2.1%) are indicated in black. The extremapoints found for 1D exploration along each singular vector are indicated by the red circles. These form the vertices of the 2-polytope.



(a) 0.5% accepted decrement ($8.2550 × 10^7)
0.6% maximum decrement ($8.2467 × 10^7)



(b) 1.0% accepted decrement ($8.2136 × 10^7)
1.1% maximum decrement ($8.2052 × 10^7)



(c) (Part 1) 2.0% accepted decrement ($8.1306 × 10^7)
3.0% maximum decrement ($8.0476 × 10^7)

Figure 5.14: 2D plane scan result - Primary direction: SV4, Secondary direction: SV2

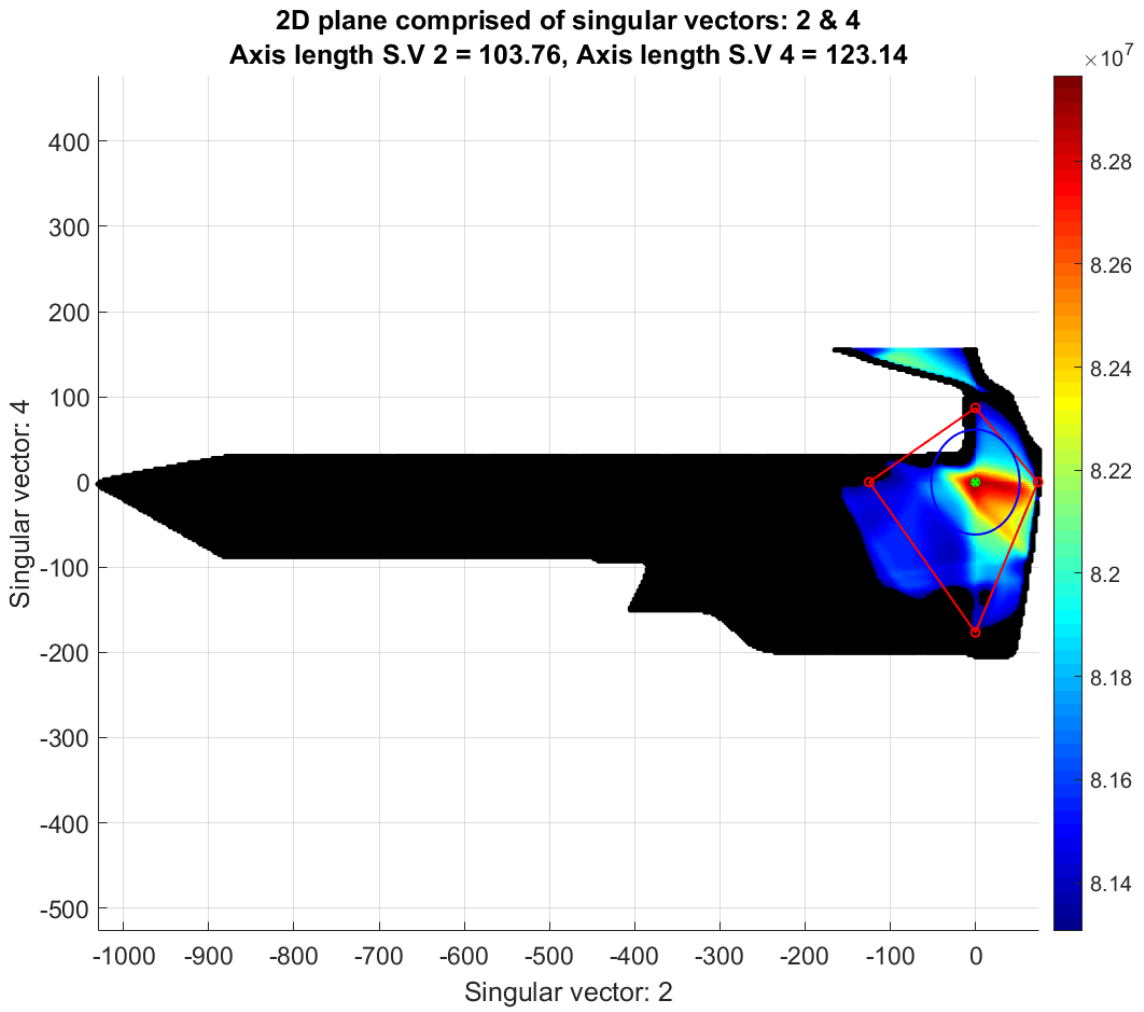


Figure 5.15: (Part 2) 2.0% accepted decrement ($\$8.1306 \times 10^7$)
3.0% maximum decrement ($\$8.0476 \times 10^7$)

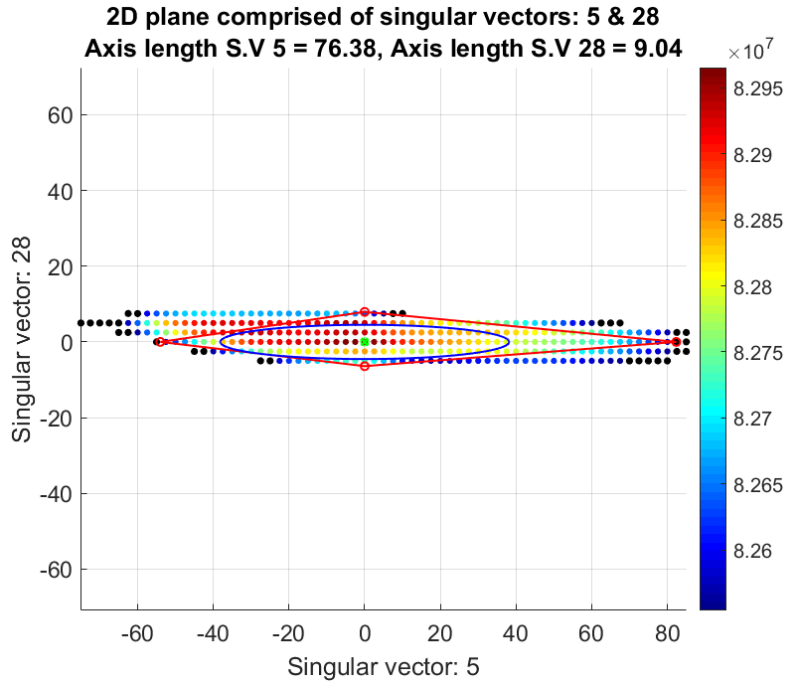
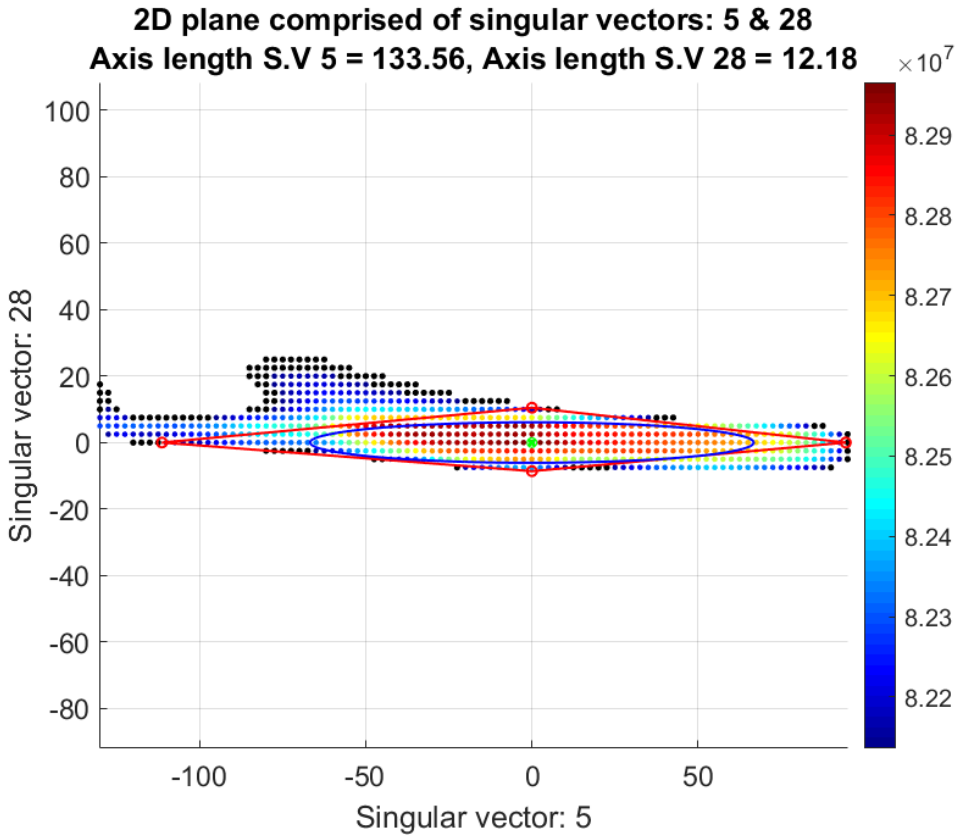
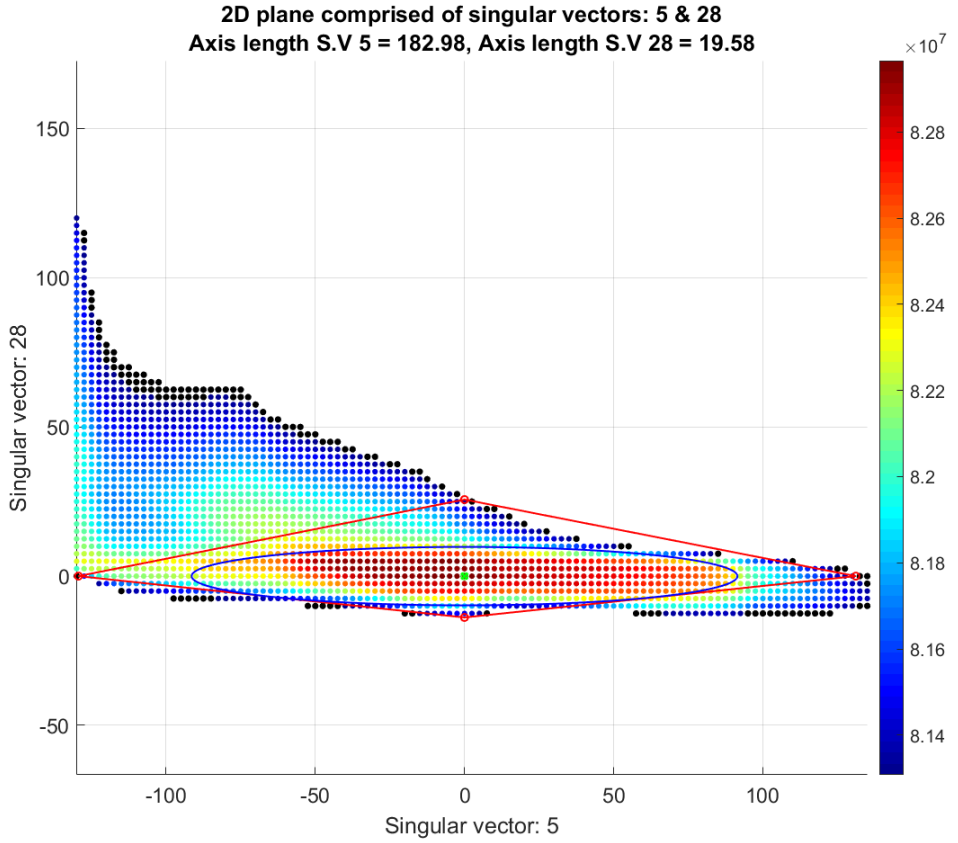


Figure 5.16: 0.5% accepted decrement ($\$8.2550 \times 10^7$)
0.6% maximum decrement ($\$8.2467 \times 10^7$)



(a) 1.0% accepted decrement ($8.2136 × 10^7$)
1.1% maximum decrement ($8.2052 × 10^7$)



(b) 2.0% accepted decrement ($8.1306 × 10^7$)
2.1% maximum decrement ($8.1223 × 10^7$)

Figure 5.17: 2D plane scan result - Primary direction: SV5, Secondary direction: SV28

Figures 5.10-5.17 lead to a few observations:

1. For plane scan results with 0.5% & 1.0% decrement relative to the optimum NPV, the inscribed 2-ellipsoid is entirely inside the feasible region. More notably even with just 4 points on the feasible region boundary, the 2-polytope (in each result) represents a reasonably accurate convex approximation of the 2D feasible region. For the 2.0% decrement case, the feasible region is clearly non-convex and sampling 4 points on the boundary is insufficient to capture this non-convexity. As a consequence, the 2-polytope in each result and the 2-ellipsoid from the P4S2 result (figure 5.14c) include non-feasible samples.

2. Since the centre for each inscribed 2-ellipsoid is fixed beforehand (as the optimum/exploration origin), the ellipsoid size is much smaller relative to the 2-polytope.

3. The arrow in figure 5.14c indicates that the feasible region is discontinuous. The disconnected region was identified only because exploration was performed till a rather large decrement (3.0%).

4. Figure 5.15 indicates that for the target function value decrement (3.0%) and singular vector direction ($-$SV2), a large exploration distance is required in the log transformed space before controls in the regular space change sufficiently. This relates back to the large magnitude of controls in the log transformed space. This result also suggests that it is sensible to impose a termination condition for exploration based on the minimum accepted change in controls in the regular space for an exploration step in the log transformed space.

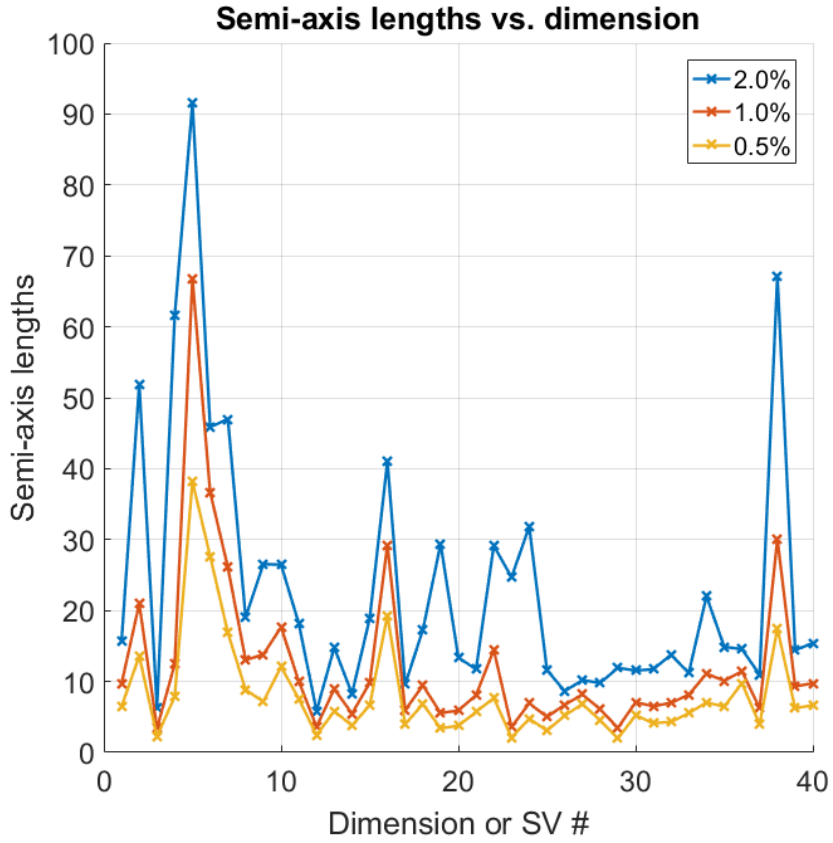


Figure 5.18: Semi-axis lengths of each d-ellipsoid for different decrement thresholds (scaling = 0)

Figure 5.18 indicates that larger accepted decrements from the optimum NPV result in longer ellipsoid semi-axis lengths. This result is similar to figure 5.9 where it was observed that larger decrements from the optimum NPV result in longer 1-dimensional exploration distances.

5.4 Validation

For a pair of singular vectors, a plane scan presents function value information along a 2-dimensional plane in d-dimensional control vector space. However, as seen in section 5.3, this information can be used to visualize the 2-ellipsoid's fit in the feasible region for that plane alone. (Recall, each 2-ellipsoid is a 2-dimensional cross-section of the d-CSC-MVIE) To test the fit quality of the d-CSC-MVIE as a whole, uniformly distributed samples are generated in this ellipsoid's interior using the algorithm detailed in Dezert et al. [8]. A second approach involving sample generation on the d-CSC-MVIE's surface is also tested. For this, the samples generated in the ellipsoid interior are projected to its surface - thereby maximizing the distance of each sample from the optimum coordinate at the ellipsoid centre. The mathematics of sample projection and results of testing the two sample generation approaches on a 3-ellipsoid are included in appendix A2.6.

In the context of this thesis, the fit quality is just a ratio that compares the number of feasible samples to the total number of samples. Each sample counts as one function evaluation and represents a unique control vector coordinate in d-dimensional space.

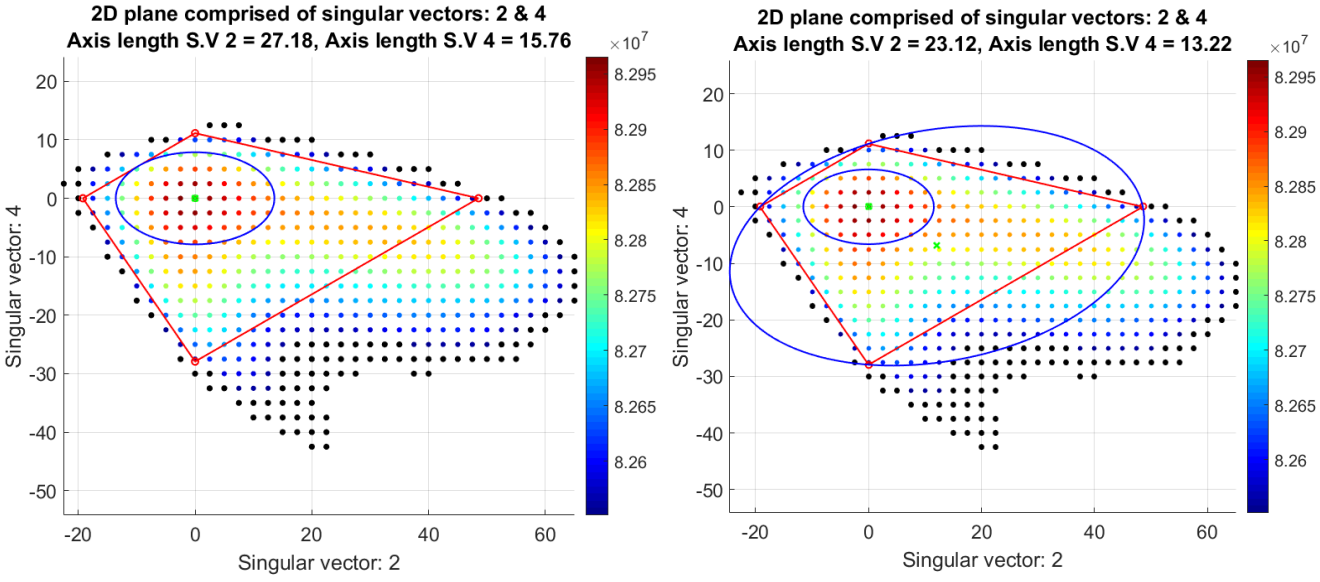
5.4.1 Ellipsoid generation

With 1-dimensional exploration, the three NPV decrement thresholds (0.5%, 1.0%, 2.0%) result in a total of three sets of $2 \times d$ extremapoints. For the purpose of validation, each set of extremapoints is used to generate three unique d-ellipsoids which include:

1. A d-CSC-MVIE with scaling coefficient = 0
2. A d-CSC-MVIE with scaling coefficient = 1
3. A d-dimensional minimum volume enclosing ellipsoid or d-MVEE (Khachiyan (1996) [28])

Each d-CSC-MVIE is generated with the ellipsoid fitting workflow described in chapter 4. Using a non-zero scaling coefficient (algorithm A2.19) produces a more conservative ellipsoid fit as seen in figure 5.19. The Matlab implementation of the Khachiyan algorithm sourced from Moshtagh (2006) [34] is used. The Khachiyan algorithm generates a d-MVEE that envelopes/encloses all $2 \times d$ extremapoints and is included here primarily to serve as the baseline for ellipsoid fit quality. A comparison of ellipsoid fitting with a 2-dimensional MVIE (maximum volume inscribed ellipsoid) and 2-MVEE is provided in appendix A2.5.

Three sets of $2 \times d$ extremapoints with three d-ellipsoids generated per set results in a total of nine unique d-ellipsoids. Each d-ellipsoid represents an approximation of the feasible region and serves as an input to the sample generation stage.



(a) 0.5% accepted decrement ($8.2550 × 10^7$)
0.6% maximum decrement ($8.2467 × 10^7$)
Scaling coefficient = 0

(b) 0.5% accepted decrement ($8.2550 × 10^7$)
0.6% maximum decrement ($8.2467 × 10^7$)
Scaling coefficient = 1, 2-MVEE included

Figure 5.19: 2D plane scan result - Primary direction: SV4, Secondary direction: SV2

Uniformly distributed samples are generated in the interior of all nine d-ellipsoids. Surface sample generation is performed specifically for the three d-CSC-MVIEs with the scaling coefficient set as zero. This results in a total of 12 tests where 10^4 samples are generated per test. (1.2×10^5 samples in total)

5.4.2 Results

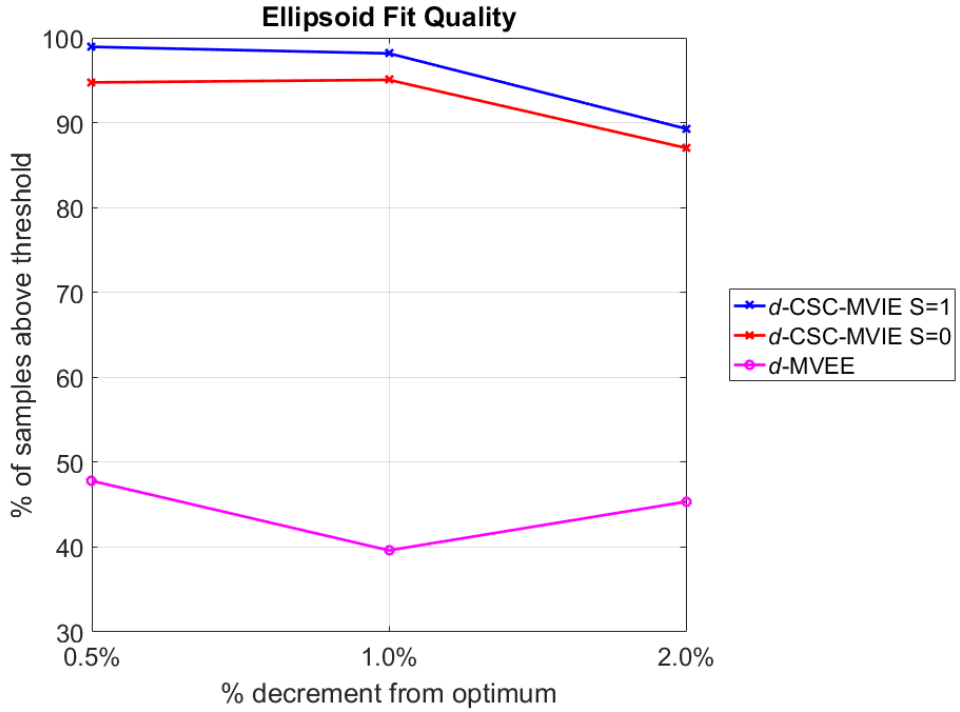


Figure 5.20: (Part 1) Ellipsoid fit results

Observations for (interior) sample generation with the d-CSC-MVIE and d-MVEE are presented first. Figure 5.20 broadens the observations in section 5.3 regarding the fit quality of a 2-ellipsoid within its 2D feasible region to the d-dimensional context.

1. For a 0.5% and 1.0% decrement relative to the optimum NPV, nearly all ($\sim 95\%$) the samples generated in the corresponding four d-CSC-MVIEs are feasible. For the two d-CSC-MVIEs associated with the 2.0% decrement threshold, the number of feasible samples decreases. This is because the feasible region boundary (as seen in figure 5.14c - in the 2-dimensional P4S2 plane scan) is now non-convex and sampling just $2 \times d$ extremapoints is insufficient to capture the non-convexity. Consequently non-feasible regions are admitted in each d-CSC-MVIE and the percentage of feasible samples decreases.

2. Setting the scaling coefficient as 1 results in a smaller ellipsoid as seen in figure 5.19. While the percentage of feasible samples is larger at each decrement threshold, a non-zero scaling coefficient decreases the number of feasible strategies admitted in each d-CSC-MVIE.

3. The number of feasible samples for each d-MVEE is $< 50\%$. This is because the Khachiyan algorithm produces a d-MVEE that envelopes or "rounds" the d-polytope comprised of a set of $2 \times d$ extremapoints. (This is presented in figure 5.19b in the two dimensional context) The extremapoints also uniquely determine the centre, axis lengths and axis orientations of this d-MVEE. In contrast, the ellipsoid fitting workflow in this thesis (chapter 4) produces a d-CSC-MVIE that is both centred at the optimum coordinate and has its cross-sections constrained by subsets of this d-polytope's vertices. This relative lack of constraints results in each d-MVEE admitting substantial non-feasible regions.

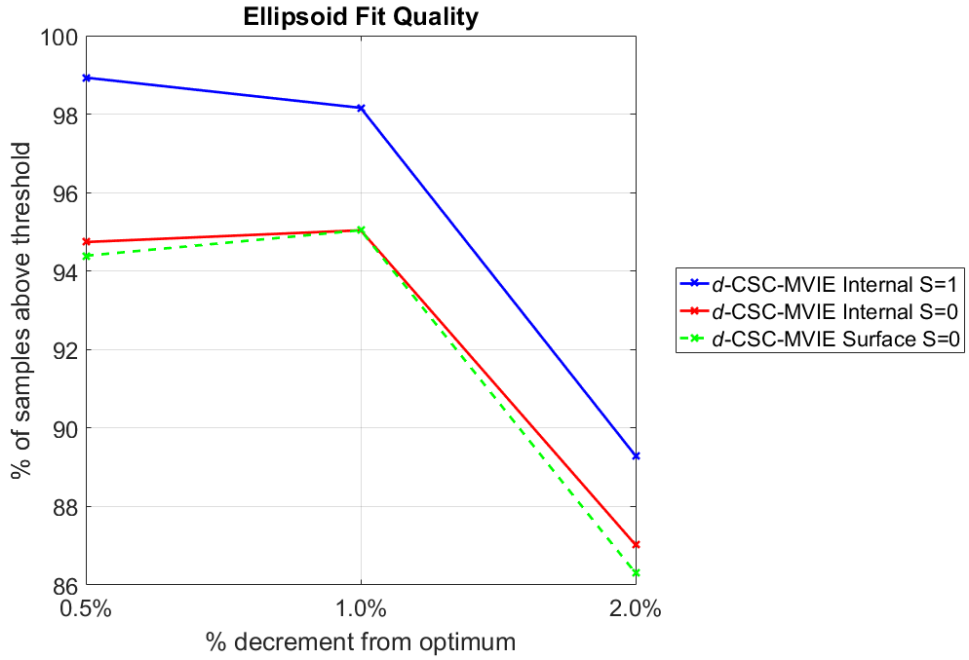


Figure 5.21: (Part 2) Ellipsoid fit results

The green dashed line in figure 5.21 indicates the number of feasible samples for each d-CSC-MVIE when sample generation is performed on the surface - which as can be seen is only marginally lesser.

Although resulting in a poor fit quality, it should be noted that the Khachiyan algorithm is versatile as it can generate an enclosing ellipsoid with as little as $d + 1$ points. Moreover, an MVEE can also enclose a randomly generated set of points. In comparison, for the CSC-MVIE ellipsoid fitting approach, it is essential that the distance from the exploration origin to each extremapoint is represented as a linear combination of an m-sized subset of singular vectors (where m is the dimension of the cross-sectional constraint polytope). This means a d-CSC-MVIE cannot be used if random sampling is performed since representation of constraints in a lower dimensional space would not be possible.

Chapter 6

Conclusions and Recommendations

6.1 Summary and conclusions

The goal of this thesis was to investigate methods to characterize feasible sets of control strategies for reservoir oil recovery. The availability of such a set of strategies can provide operators with flexibility to meet multiple objectives and constraints and also the potential to adapt to changing circumstances without deviating from the main (economic) objective. The main challenge posed by reservoir control problems is that the dimensionality of the problems (the number of controls) may be very large, in the order of hundreds controls.

The multiple ellipsoid based sampling (MEBS) approach in Zamora-Sillero et al. (2011) [44] was initially investigated. A few limitations were identified regarding the application of this approach to reservoir problems:

1. Ellipsoid expansion/contraction is based on uniform (random) sampling which while certainly thorough is inefficient in the context of a high dimensional non-convex feasible region.
2. The feasible samples generated by the MEBS workflow (rather than the enclosing ellipsoids) are used to characterize the feasible region. An ellipsoid enclosing a set of feasible samples can itself include non-feasible regions. Each ellipsoid serves only as a "container" for a subset of feasible samples found.

A precise characterization of the feasible region through MEBS while ideal is not computationally tractable for the reservoir case. Analysis of the MEBS approach did however lead to the idea that the feasible region can be approximately characterized with an ellipsoid.

For many reservoir optimization problems the objective function in the neighborhood of the optimum can be described as a nearly flat (ridge or plateau shaped) region within which the objective function value hardly decreases. To find a set of control strategies that characterizes such a region an ensemble-based optimization approach is investigated where the BFGS scheme iteratively builds up information about the curvature of the objective function space around one, possibly local, optimal solution. The main conclusions of the optimization step are as follows:

1. An optimal control strategy that maximizes the objective function was found. ($\$ 8.2965 \times 10^7$)
2. Due to its approximate nature, the norm of the EnOpt gradient vector never truly reaches zero.
3. The use of strong Wolfe line search ensures that the BFGS Hessian is positive definite at each iterate. However, testing for the "curvature" condition with an EnOpt gradient is computationally expensive since the gradient must be computed at multiple trial iterates of the line search in one optimization iteration.
4. The number of BFGS updates (62) exceeds the number of controls in the reservoir model ($d = 40$). This indicates the BFGS Hessian has sufficiently resolved curvature of the objective function space around the optimum.

The second step investigated in this thesis is the exploration of the objective function space around the optimum. The null space of the Hessian matrix as characterized by its singular vectors provides search directions that are explored using a line search. The boundary of the feasible region is sampled at $2 \times d$ extremapoints (end points of each line search). The main findings of the exploration step are as follows:

1. Each extremapoint represents a unique control strategy but with an identical objective function value.
2. The singular values of the BFGS Hessian do not provide a provide a good indication of the explorable distance in the feasible region along the corresponding singular vector directions.

The third step investigated in this thesis is the characterization of the identified set of extremapoints. Initially, this thesis considered the use of a d-MVEE based on the Khachiyan algorithm [28] to enclose the $2 \times d$ extremapoints - as a direct approximation of the feasible region. Tests on 2D plane scans (figure 5.19b) indicated however that even a 2-MVEE would certainly include non-feasible regions. As an alternative, a d-MVIE was considered, which is an ellipsoid inscribed entirely within the d-polytope comprised of $2 \times d$ extremapoints. However, it was observed that the number of constraints (d-polytope facets) scaled exponentially in relation to the number of dimensions. As a result, this approach became computationally intractable.

We therefore propose as a solution, a d-CSC-MVIE or d-dimensional cross-section constrained maximum volume inscribed ellipsoid fitting workflow. This workflow based in convex optimization [19] [20] generates an inscribed ellipsoid that is centred at the optimum and offers some flexibility in terms of control strategies, while ensuring the objective function value is still mostly above a specified threshold. By constraining only cross-sections, the number of constraints is made tractable and the flexibility of a d-MVIE in fixing the centre coordinate is preserved. This workflow represents a combination of ideas presented in Director & Hatchel (1977) [9] and Karl et. al. (1994) [27].

Validation testing (figure 5.21) found $\sim 95\%$ of samples to lie above the function value threshold till a 1.0% decrement from the optimum NPV. By using a d-CSC-MVIE and therefore limiting consideration to the immediate vicinity of the optimum, the requirement for extensive function evaluations in exploration is sidestepped. Though it should be noted, better exploration will result in a larger number of strategies admitted in the d-CSC-MVIE.

Finally, the proposed workflow was demonstrated on a small-size reservoir optimization problem containing 5 wells. A set of optimal control strategies for the 5 wells could be obtained suggesting a continuum of fairly different strategies indicating the scope for significant flexibility.

6.2 Recommendations

1. In this thesis a deterministic problem was considered. In reality one cannot assume that any single reservoir model will provide exact predictions, that is, predictions that exactly match results obtained from the true reservoir. The uncertainty in subsurface geology can be represented by an ensemble of geological models. Optimization of the expected objective function is often referred to as robust optimization. Follow-up work should consider methods to characterize the expected objective function space in an efficient manner.

2. It was observed that the singular values of the Hessian (as estimated from ensemble optimization with the BFGS scheme) do not provide a good indication of the extent of the feasible region along the corresponding singular vector directions. Consequently, the Hessian estimation step may not be necessary. This can be investigated further. Use of adjoint gradients could for example help verify the result obtained with ensemble gradients.

3. This thesis provides a method that characterizes a set of feasible strategies defined by a limit in the decrease in the objective function value relative to the optimum. However, more research is needed to exploit this (ellipsoidal) solution set in order to identify solutions (i.e., control strategies) that optimize additional objectives (e.g., short term production targets). Furthermore, it is desirable to identify solutions that provide the flexibility for an operator to switch operating strategies midway during production if circumstances demand it.

4. Alternative approaches that deal with the problem of flexibility in operating strategy have been proposed in the past (e.g., Van Essen et al. (2009) [39]). A more complete comparison of such approaches with the one proposed in this thesis would be valuable.

5. The computational complexity of the CSC-MVIE algorithm (appendix A2.19) is yet to be assessed, particularly as a function of the choice in m and d. The scaling coefficient which allows for a more conservative ellipsoid fit is uniquely configurable for each of the ${}^{d}C_{2}$ 2-planes. Tuning this coefficient on a per plane basis may result in a larger number of feasible strategies being admitted in the d-CSC-MVIE.

6. Controls in the log transformed space associated with controls at bounds in the regular space attain large magnitudes. This is a hindrance to exploration since only few controls actually undergo change. Imposing a temporal correlation with a moving average or with time series smoothing should be investigated as this can prevent any one control from becoming excessively large. Alternatively, imposing upper/lower bounds on the optimal control vector in the log transformed space prior to exploration should also be investigated.

Appendices

Appendix A1

Optimization

A1.1 Mathematics of the Rosenbrock function

The generalized Rosenbrock equation for d-dimensional input vector $\boldsymbol{x}$ is:

$$J(\boldsymbol{x}) = \sum_{i=1}^{d-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \tag{A1.1}$$

The optimum coordinate for is always a d-dimensional vector of ones where the function value is 0.

A1.1.1 Analytical gradient computation

The $d \times 1$ gradient vector when $d > 2$ is computed as:

$$\begin{aligned}
&\frac{\partial J}{\partial x_1} = 400x_1^3 - 400x_1x_2 + 2x_1 - 2 \\
&\frac{\partial J}{\partial x_d} = -200x_{d-1}^2 + 200x_d \\
&\textbf{for } i \in \{2, \cdots, d-1\} \\
&\qquad \frac{\partial J}{\partial x_i} = 202x_i - 2 + 400x_i^3 - 400x_ix_{i+1} - 200x_{i-1}^2 \\
&\text{end } \textbf{for}
\end{aligned} \tag{A1.2}$$

A1.1.2 Analytical Hessian computation

The $d \times d$ Hessian matrix when $d > 2$ is computed as:

$$\begin{aligned}
&\frac{\partial^2 J}{\partial x_1^2} = 1200x_1^2 - 400x_2 + 2 \quad \frac{\partial^2 J}{\partial x_1 \partial x_2} = -400x_1 \\
&\frac{\partial^2 J}{\partial x_d \partial x_{d-1}} = -400x_{d-1} \qquad \frac{\partial^2 J}{\partial x_d^2} = 200 \\
&\textbf{for } i \in \{2, \cdots, d-1\} \\
&\quad \textbf{for } j \in \{1, \cdots, d\} \\
&\qquad \textbf{if } j < i-1, \qquad \frac{\partial^2 J}{\partial x_i \partial x_j} = 0 \\
&\qquad \textbf{if } j = i-1, \qquad \frac{\partial^2 J}{\partial x_i \partial x_j} = -400x_{i-1} \\
&\qquad \textbf{if } j = i, \qquad \frac{\partial^2 J}{\partial x_i \partial x_j} = 1200x_i^2 - 400x_{i+1} + 202 \\
&\qquad \textbf{if } j = i+1, \qquad \frac{\partial^2 J}{\partial x_i \partial x_j} = -400x_i \\
&\qquad \textbf{if } j > i+1, \qquad \frac{\partial^2 J}{\partial x_i \partial x_j} = 0 \\
&\quad \text{end } \textbf{for} \\
&\text{end } \textbf{for}
\end{aligned} \tag{A1.3}$$

When $d = 2$, the **for** loop structures can be ignored in both gradient and Hessian computation.

A1.2 Trial Step Length Calculation

The process of calculating the trial step length for the steepest ascent (Reservoir case) and steepest descent (Rosenbrock model) algorithms are presented below:

If the second derivatives of a d-dimensional objective function J are bounded and continuous, a truncated Taylor series expansion can be used to express the change in function value arising due to step α along a search direction $\boldsymbol{p}_k$ in terms of derivatives:

$$J(\boldsymbol{u}_k + \alpha \boldsymbol{p}_k) \approx J(\boldsymbol{u}_k) + \alpha(\nabla J(\boldsymbol{u}_k))^T \boldsymbol{p}_k + \mathcal{O}(\|\Delta \boldsymbol{u}\|^2), \tag{A1.4}$$

where $\nabla J(\boldsymbol{u}_k)$ is the $d \times 1$ gradient vector. Ignoring the error term $\mathcal{O}(\|\Delta \boldsymbol{u}\|^2)$, the first order approximation of the above equation becomes:

$$J(\boldsymbol{u}_k + \alpha \boldsymbol{p}_k) = J(\boldsymbol{u}_k) + \alpha(\nabla J(\boldsymbol{u}_k))^T \boldsymbol{p}_k. \tag{A1.5}$$

Since $\boldsymbol{u}_k + \alpha \boldsymbol{p}_k$ is the coordinate at iteration $k+1$, equation A1.5 can be rewritten as:

$$J_{k+1} = J_k + \alpha(\nabla J(\boldsymbol{u}_k))^T \boldsymbol{p}_k. \tag{A1.6}$$

The product $(\nabla J(\boldsymbol{u}_k))^T \boldsymbol{p}_k$ in A1.5 is the directional derivative. This represents the first order approximation derived change in function value along direction $\boldsymbol{p}_k$ for a given step length. Knowing this, it is possible to calculate a trial step length by limiting the acceptable change in objective function value. If $J_{k+1} - J_k = \Delta J$, one can impose:

$$\Delta J \leqslant c_{\text{slo}}\, J_k, \tag{A1.7}$$

which becomes,

$$\alpha_{\text{slo}}(\nabla J(\boldsymbol{u}_k))^T \boldsymbol{p}_k \leqslant c_{\text{slo}}\, J_k. \tag{A1.8}$$

Therefore, the trial step length α_{slo} is calculated as:

$$\alpha_{\text{slo}} \leqslant \frac{c_{\text{slo}}\, J_k}{(\nabla J(\boldsymbol{u}_k))^T \boldsymbol{p}_k}. \tag{A1.9}$$

Alternatively, a restriction can be imposed on the maximum change in the norm of the control vector $\boldsymbol{u}_k$.

If

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \alpha \boldsymbol{p}_k, \tag{A1.10}$$

Then,

$$\boldsymbol{u}_{k+1} - \boldsymbol{u}_k = \Delta \boldsymbol{u} = \alpha \boldsymbol{p}_k. \tag{A1.11}$$

One can impose the requirement that,

$$\|\Delta \boldsymbol{u}\| \leqslant c_{\text{slc}}\, \|\boldsymbol{u}_k\|, \tag{A1.12}$$

Which is equivalent to,

$$\alpha_{\text{slc}}\, \|\boldsymbol{p}_k\| \leqslant c_{\text{slc}}\, \|\boldsymbol{u}_k\|. \tag{A1.13}$$

Therefore, the trial step length α_{slc} is calculated as:

$$\alpha_{\text{slc}} \leqslant \frac{c_{\text{slc}}\, \|\boldsymbol{u}_k\|}{\|\boldsymbol{p}_k\|}. \tag{A1.14}$$

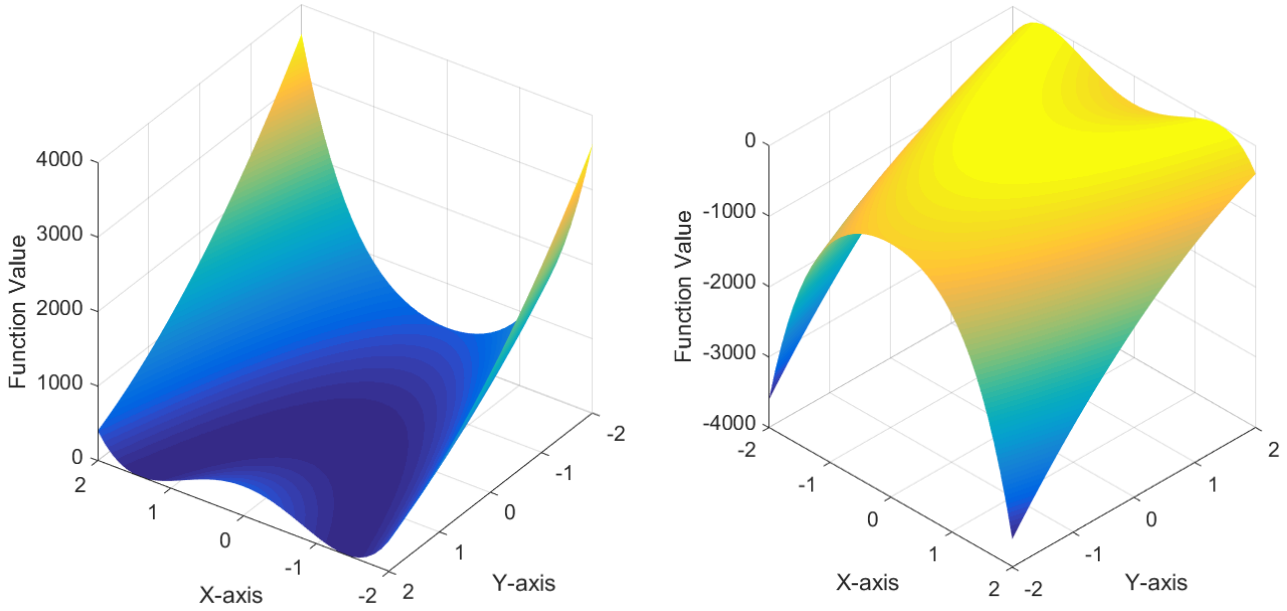
The most restrictive step length is finally chosen as:

$$\alpha = \min(\alpha_{\text{slo}}, \alpha_{\text{slc}}). \tag{A1.15}$$

In the regular Rosenbrock model where steepest descent optimization is demonstrated, $c_{\text{slo}} = -0.25$ and $c_{\text{slc}} = 0.1$. The negative sign indicates the function value should decrease along the search direction. For the reservoir case, $c_{\text{slo}} = 0.1$ and $c_{\text{slc}} = 1.0$

A1.3 Validating the BFGS implementation

Before implementing the BFGS scheme on the reservoir model it is sensible to test if the implementation works correctly on a simpler toy model. The Rosenbrock function is computationally inexpensive to evaluate - this fact combined with the ready availability of analytical first and second order derivative information make it ideal in validating the BFGS scheme.



(a) Regular 2D Rosenbrock

(b) Inverted 2D Rosenbrock

Figure A1.1: Surface plot of the 2D Rosenbrock function

A1.3.1 Initial optimizer settings

Initial coordinate	$(-1.3, 1.4)$
Max iterations	10000
Absolute function value	10^{-16}
Gradient norm	10^{-12}

Table A1.1: Common criteria - Last two entries indicate stopping conditions

Inner iterations	100
Contraction factor	0.5

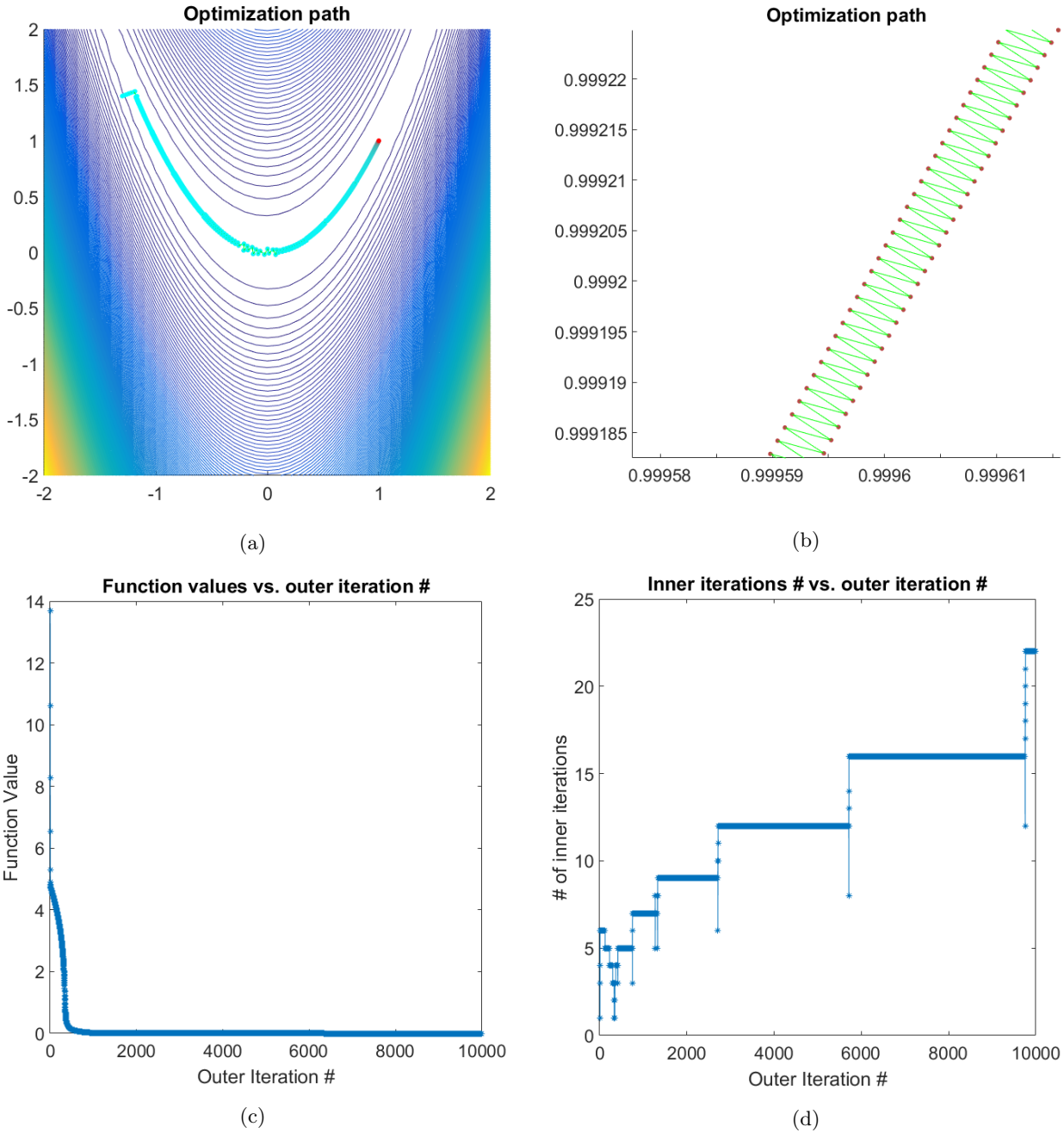
Table A1.2: Armijo backtracking line search settings

Wolfe c_1	10^{-4}
Wolfe c_2	0.9
Wolfe iterations	5
Zoom iterations	50
Trial step length	1

Table A1.3: Strong Wolfe line search settings

A1.3.2 Validation on 2D Rosenbrock function (Steepest Descent)

The results for optimization using steepest descent are provided first. This serves as the baseline in optimization performance. Armijo backtracking line search is used and inner iterations represent the number of times backtracking (step length contraction) is performed until an acceptable length is found. The characteristic zig-zag optimization path in the Rosenbrock valley can also be observed.



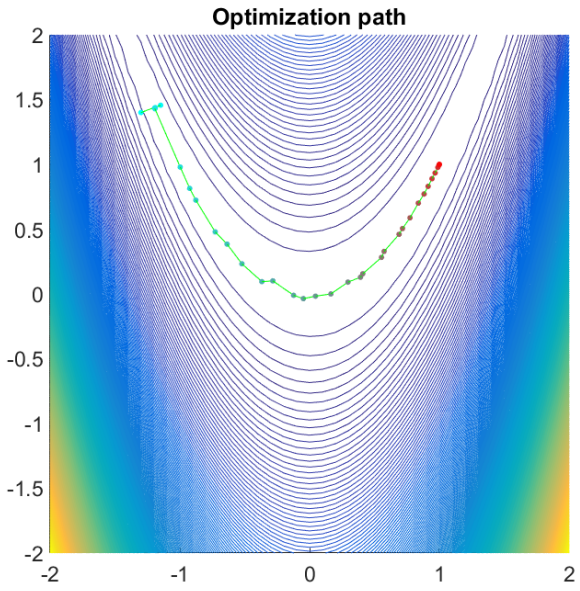
(a) (b) (c) (d)

Figure A1.2: Steepest descent results

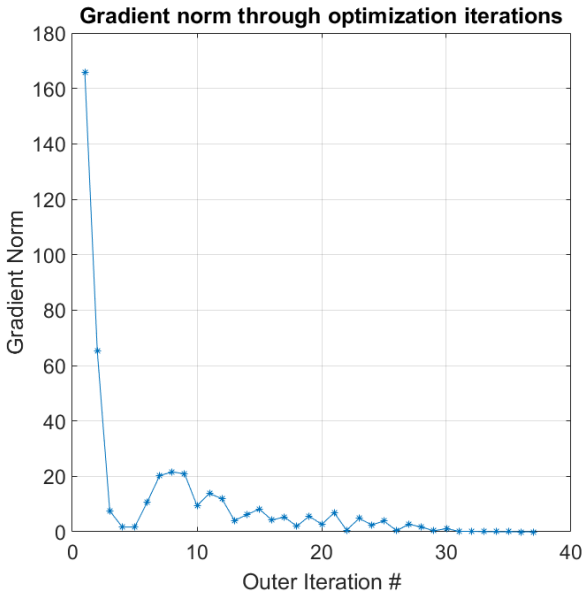
Optimization Iterations	10000
Function Evaluations	136091
Gradient Evaluations	10000
Final function value	4.2751×10^{-10}
Final gradient norm	3.3764×10^{-5}
Stopping Criteria Met?	**NO**

Table A1.4: Steepest descent results

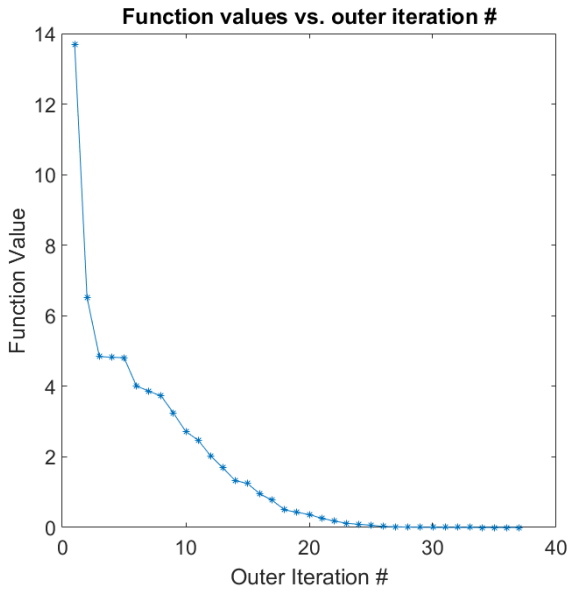
A1.3.3 Validation on 2D Rosenbrock function (Quasi-Newton)



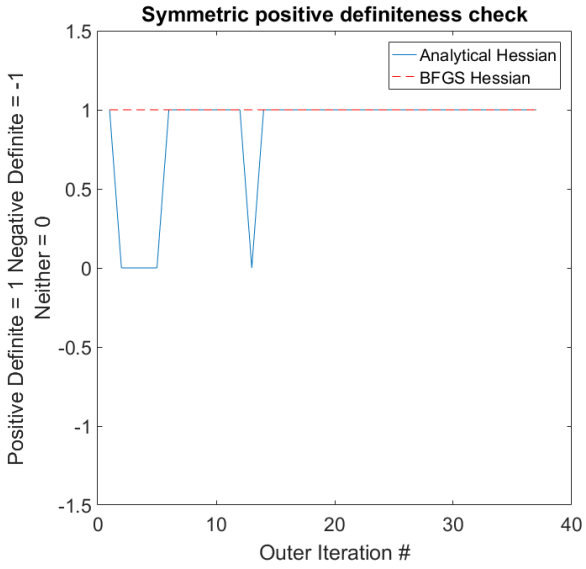
(a)



(b)



(c)



(d)

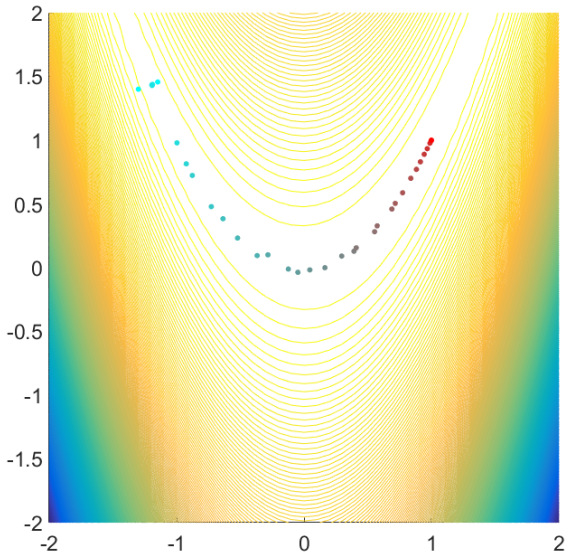
Optimization Iterations	37
Function Evaluations	125
Gradient Evaluations	101
Final function value	1.5529×10^{-22}
Final gradient norm	5.3747×10^{-10}
Stopping Criteria Met?	**YES**

Table A1.5: Quasi-Newton results - 2D Rosenbrock

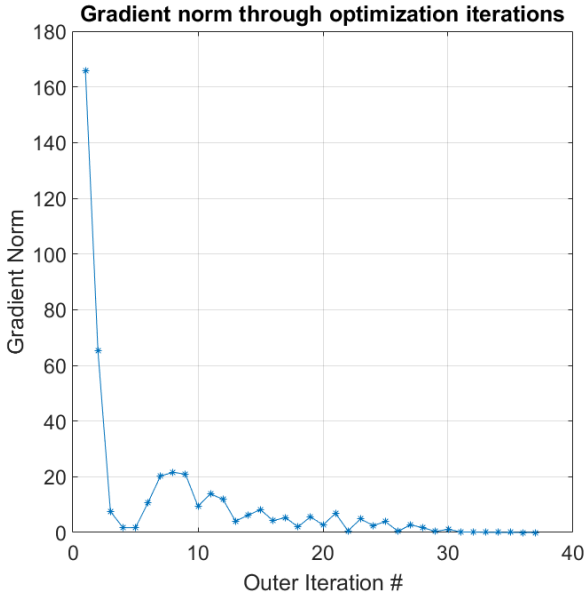
Comparing tables A1.4 and A1.5, quasi-newton optimization requires almost 3 orders of magnitude fewer function evaluations to find the optimum. A1.3d presents the matrix eigen value test where the BFGS Hessian is always positive definite - regardless of the analytical Hessian. This follows from the nature of the BFGS update. (Nocedal & Wright (2006) [36]) The two Hessians are also nearly identical at the last optimization iteration.

$$\boldsymbol{H}_{\text{BFGS}} = \begin{bmatrix} 801.7217 & -399.8517 \\ -399.8517 & 199.9210 \end{bmatrix} \qquad \boldsymbol{H}_{\text{Analytical}} = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}$$

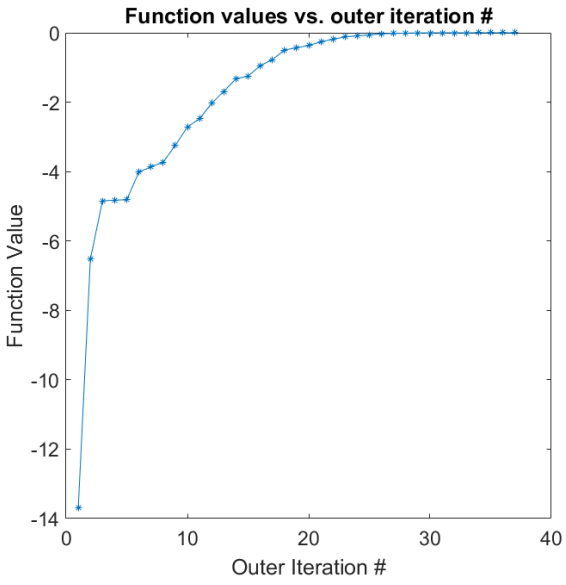
A1.3.4 Validation on inverted 2D Rosenbrock function (Quasi-Newton)



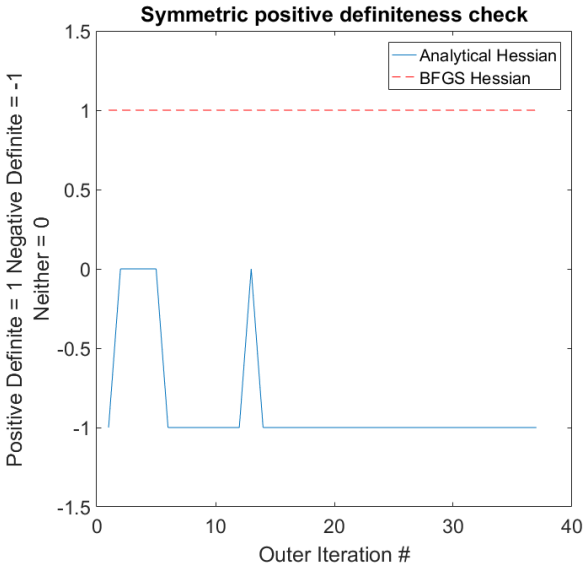
(a)



(b)



(c)



(d)

Optimization Iterations	37
Function Evaluations	125
Gradient Evaluations	101
Final function value	-1.5529×10^{-22}
Final gradient norm	5.3747×10^{-10}
Stopping Criteria Met?	**YES**

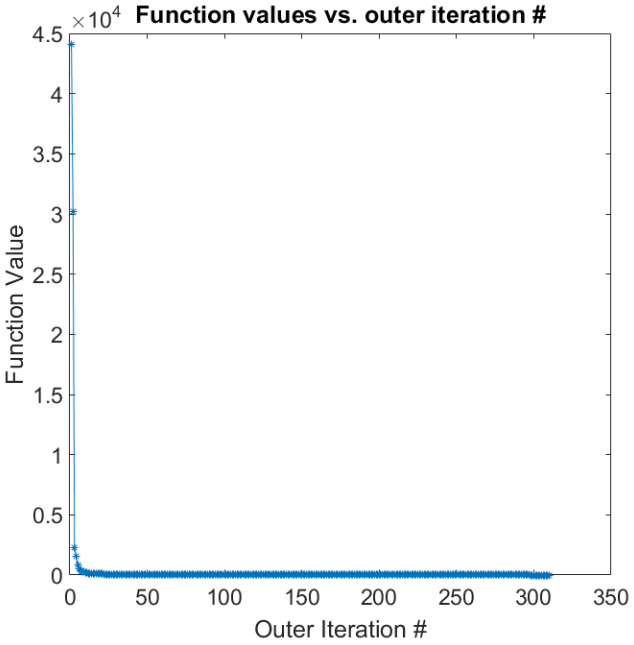
Table A1.6: Quasi-Newton results - Inverted 2D Rosenbrock

The inverted Rosenbrock function is defined mathematically by adding a negative sign to each: objective function value, gradient and Hessian result. The main goal of this test is to verify that sign conventions have been followed appropriately from the perspective of a maximization problem. Below, the matrix negative signs are flipped since the BFGS Hessian is positive definite while the analytical Hessian is negative definite.

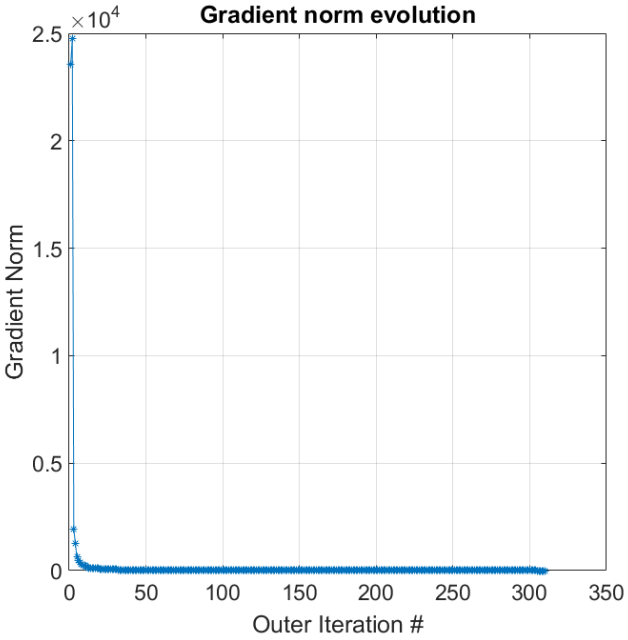
$$\boldsymbol{H}_{\text{BFGS}} = \begin{bmatrix} 801.7217 & -399.8517 \\ -399.8517 & 199.9210 \end{bmatrix} \qquad \boldsymbol{H}_{\text{Analytical}} = \begin{bmatrix} -802 & 400 \\ 400 & -200 \end{bmatrix}$$

A1.3.5 40D Rosenbrock function

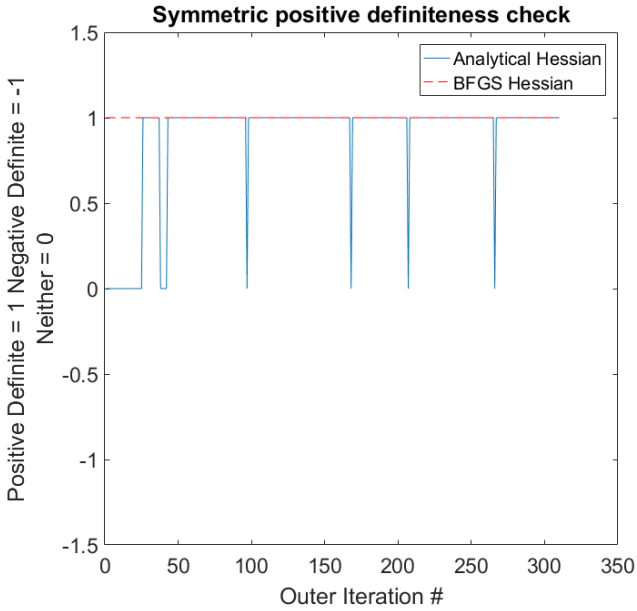
The goal of this test is to generate a BFGS Hessian whose singular vectors can be used to explore the Rosenbrock function's feasible region. The explorable distances with singular vectors from this BFGS Hessian are compared against explorable distances using singular vectors directly from the analytical Hessian at the optimum. The initial coordinate is a random 40D vector - generated in MATLAB with the Mersenne Twister random number generator: `rng(0,'twister')`



(a)



(b)



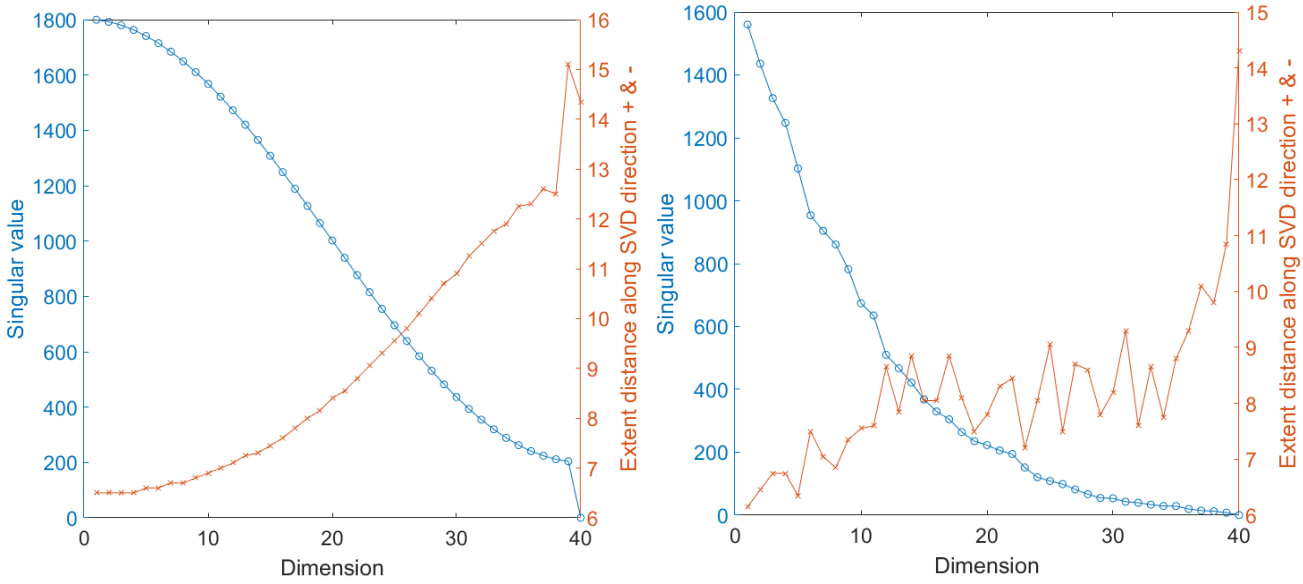
(c)

Optimization Iterations	310
Function Evaluations	687
Gradient Evaluations	653
Final function value	6.9444×10^{-17}
Final gradient norm	3.5475×10^{-7}
Stopping Criteria Met?	**YES**

Table A1.7: Quasi-Newton results - 40D Rosenbrock

Explorable distances - 40D Rosenbrock case

As with optimization, exploration was first validated on the Rosenbrock function. Figure A1.6 presents exploration results with the analytical and BFGS Hessian in the 40D Rosenbrock case (from the optimum coordinate).



(a) Exploration with analytical Hessian at optimum

(b) Exploration with BFGS Hessian at optimum

Figure A1.6: Singular value magnitude vs. total explorable distance - 40D Rosenbrock case

In figure A1.6 singular value magnitudes are presented on the left (regular) Y-axis while explorable distances for each singular vector are presented on the right Y-axis. The explorable distance for a given singular vector is measured as the euclidean distance between its two extremapoints in regular Cartesian space. Exploration is terminated when a function value of 10^4 is reached and the optimum itself has a function value of zero.

Compared to the more realistic reservoir case in section 5.2, the explorable distances for the 40D Rosenbrock function are indeed inversely related to singular value magnitudes. This is clearly seen in exploration with the analytical Hessian (figure A1.6a) where explorable distances increase near monotonically with decrease in singular value. Exploration with the BFGS Hessian (figure A1.6b) also produces a similar result but the trend is not as clear.

A1.4 Optimization With Log Transformed Controls

As described in section 2.3.1, the EnOpt gradient requires computing a perturbed ensemble of controls. The non-linearity of the log transformed space poses a few challenges to this process and the attempts made in developing a working optimizer are presented below.

Background

The TNO EnOpt workflow written in MATLAB is used by this thesis and the workflow is primarily intended for optimization in the regular control space. For an input control vector $\boldsymbol{u}_i$ the corresponding EnOpt gradient vector $\boldsymbol{g}_i$ can be computed as:

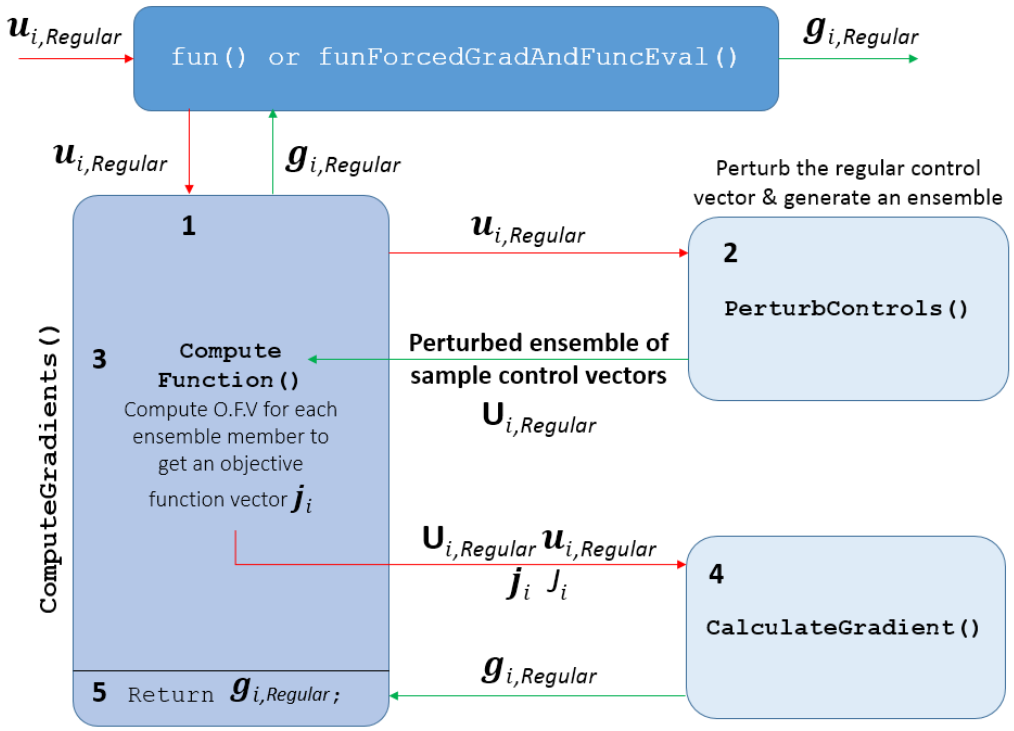


Figure A1.7: Gradient computation workflow for an input vector in regular control space. Numbers indicate sequence of steps. The subscript "*Regular*" has been added to indicate computation in regular control space.

1. A control vector $\boldsymbol{u}_{i,Regular}$ enters the function `ComputeGradients()` wherein it is passed to `PerturbControls()`
2. In `PerturbControls()`, an ensemble of perturbed control vectors is generated - as described in section 2.3.1 - and stored along the columns of matrix $\mathbf{U}_{i,Regular}$ - which is returned to `ComputeGradients()`
3. `ComputeFunction()` is then used to determine the objective function value of each ensemble member which is stored in $\boldsymbol{j}_i$
4. With the initial control vector $\boldsymbol{u}_{i,Regular}$, perturbed ensemble of controls $\mathbf{U}_{i,Regular}$ and corresponding objective function values in J_i and $\boldsymbol{j}_i$, the gradient is computed as described in equation 2.10.
5. The gradient vector $\boldsymbol{g}_{i,Regular}$ is returned to the calling program

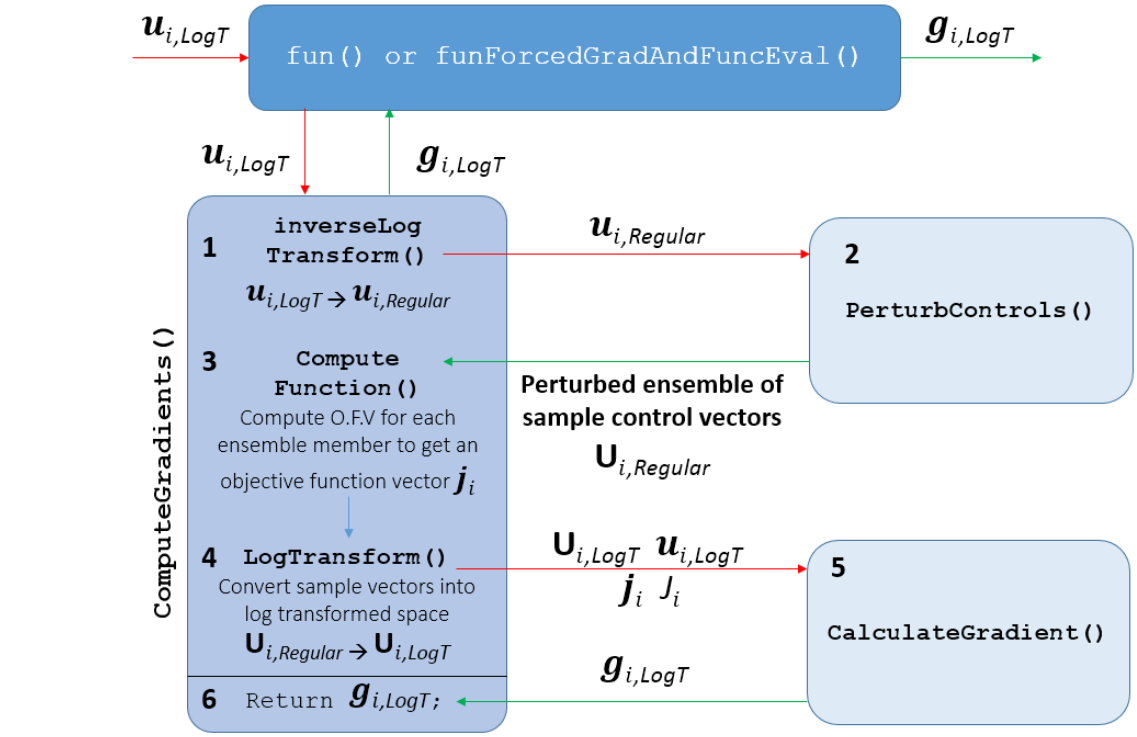
The above workflow serves as reference against which the log transformed workflow can be compared.

Impact of non-linearity on perturbations

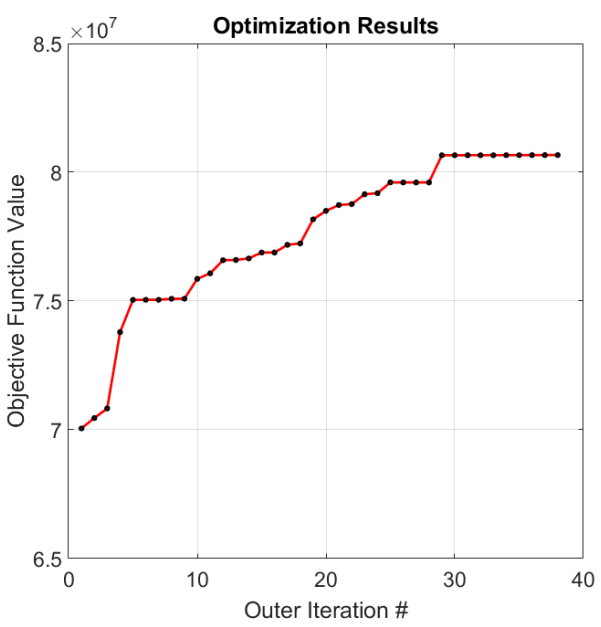
As can be seen in figure 3.2, a small perturbation on a control in the log transformed space will result in near zero change to the control in regular space - if it is near the bounds. A negligible perturbation typically translates to zero change in objective function value and results in zero gradient.

Wang et al. (2007) [42] identify this issue and suggest computing the perturbation size for a log transformed control based on a fixed perturbation size to the regular control itself. This approach has been incorporated into the TNO workflow and the updated sequence of steps for gradient computation are presented in figure A1.8

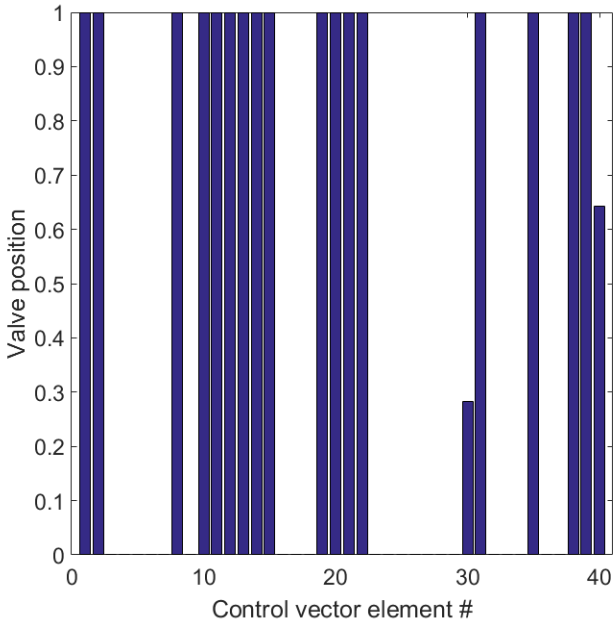
Log transformed workflow



(a)



(b) Objective function evolution



(c) Controls at iteration 38

Figure A1.8: An NPV of \$8.0669 $\times 10^7$ is reached at 38 iterations The same initial conditions are used as described in section 5.1

1. A control vector $\boldsymbol{u}_{i,LogT}$ enters the function `ComputeGradients()` wherein it is first converted to the regular control space using `inverseLogTransform()` and is then passed to `PerturbControls()`
2. In `PerturbControls()`, an ensemble of perturbed control vectors is generated - as described in section 2.3.1 - and stored along the columns of matrix $\mathbf{U}_{i,Regular}$ - which is returned to `ComputeGradients()`
3. `ComputeFunction()` is now used to determine the objective function value of each ensemble member which is stored in $\boldsymbol{j}_i$
4. At this point `LogTransform()` is used to convert each perturbed control vector back into the log transformed space as $\mathbf{U}_{i,LogT}$

5. With the initial control vector $\boldsymbol{u}_{i,LogT}$, perturbed ensemble of controls $\mathbf{U}_{i,LogT}$ and corresponding objective function values in J_i and $\boldsymbol{j}_i$, the gradient is computed (now with respect to the log transformed control vector) as described in equation 2.10.
6. The gradient vector $\boldsymbol{g}_{i,LogT}$ is returned to the calling program

By generating the perturbed ensemble directly in the regular control space the issue of perturbation non-linearity is eliminated. It should be noted that the approach followed by this thesis perhaps deviates slightly from Wang et al. (2007) [42] since they suggest using the regular space only to identify the size of the perturbation for a log transformed control.

In practice, the optimization performance using this workflow was not satisfactory as:
1. Optimum NPV was poor and nearly all controls in the final few iterations were at bounds
2. Elements of the gradient vector corresponding to controls near bounds were often substantially larger in magnitude relative to other elements.

While a Bang-Bang type solution is possible (Wang et al. (2007) [42]), neither behavior was observed when testing optimization in regular control space. Investigating further the problem was identified as the spread in values of a control across the log transformed ensemble - when that control is near bounds in regular space.

Explanation: When a control vector $\boldsymbol{u}$ is subjected to perturbation it is possible for a control u_i, which is near bounds to either become smaller than the lower bound u^{min} or become larger than the upper bound u^{max}. In both cases, the control has to be reset to the lower or upper bounds respectively. Recall from section 3.2.1 that controls at bounds in the regular space become large in the log transformed space - reaching ± 36.0437. Since an ensemble of control vectors is generated - each with a unique perturbation - it is possible that the same control in a second vector of the ensemble has been perturbed by a much smaller magnitude wherein such a reset is not required - as it is still inside bounds. When the log transform is applied to the control vector ensemble the control in the first ensemble vector attains a large magnitude while the same control in the second vector is relatively smaller. Such an unnaturally large spread in values for a specific control across the control vector ensemble results in the corresponding element of the gradient vector becoming very large magnitude wise - which most likely results in the gradient becoming a poor search direction. To address this issue, the controls in the perturbed ensemble $\mathbf{U}_{i,Regular}$ are heuristically limited to a minimum of 10^{-4} or maximum of 0.9999 prior to applying the log transform. These values lie within the bound constraints $u^{min} = 10^{-6}$ and $u^{max} = 1$ specified in equation 2.2 and consequently the spread in values of a control across the log transformed ensemble is reduced. The final log transformed workflow therefore becomes:

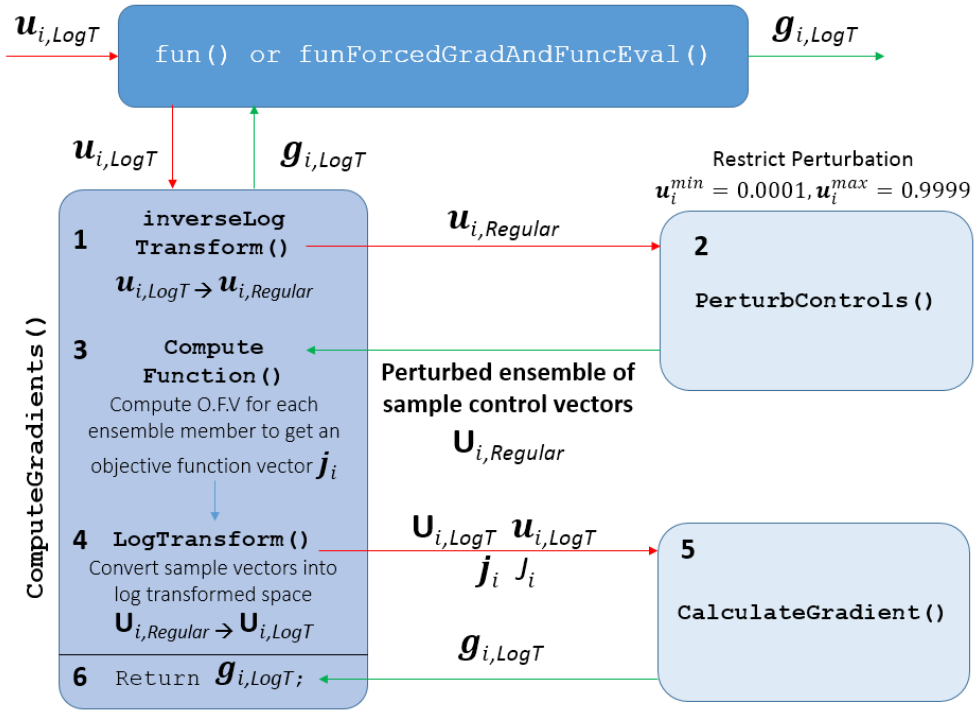


Figure A1.9: Except for the restricted perturbations, this workflow is identical to figure A1.8a. Refer to figure 5.1 for optimization results using this workflow

Appendix A2

Inscribing an ellipsoid & validation

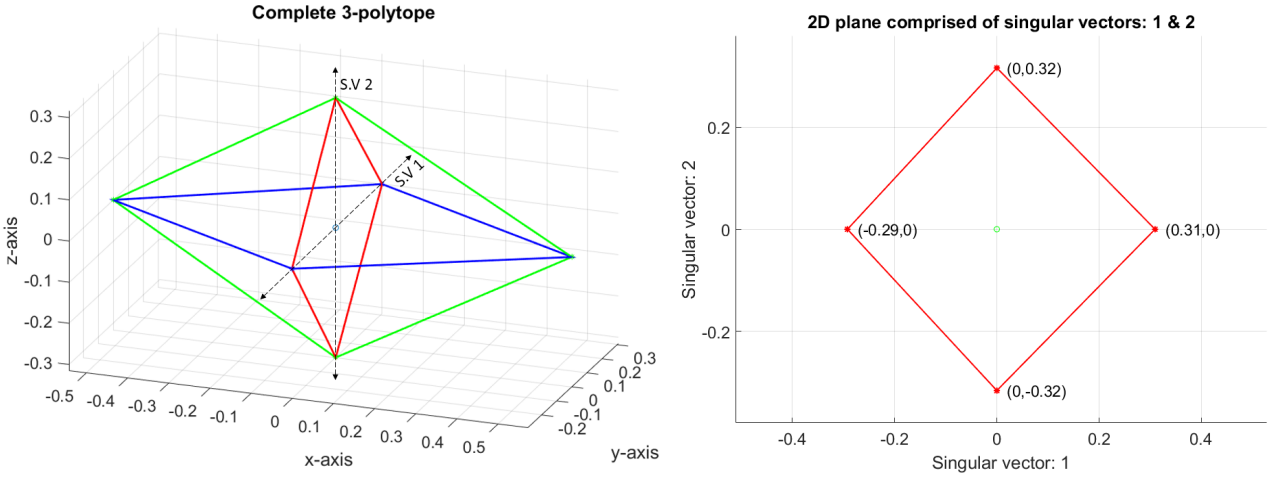
A2.1 Converting vertices in d-dimensions to vertices in m-dimensions

The vertices of each m-polytope needs to be defined. Recollect that singular vectors are used as search directions during exploration. With m selected singular vectors, only the subset of extremapoints in d-dimensional space that lie in the span of these m vectors can be considered as vertices for the m-polytope. The remaining extremapoints become unreachable with the chosen set of vectors.

One may note the extremapoints are initially coordinates in d-dimensional space. These need to be represented in an m-dimensional space. The distance from the optimum to each extremapoint in d-dimensions can be expressed as some linear combination of the singular vectors. For a chosen extremapoint a change of basis matrix assigns each singular vector (out of d total vectors) a coefficient that indicates its contribution to this distance. This way a distance can be expressed with singular vectors as components instead of Cartesian axes.

Since our consideration is limited to the extremapoints that lie in the space spanned by m singular vectors, the coefficients for the $d - m$ vectors will be zero. Amongst the remaining m coefficients, non-zero values will be assigned only to those singular vectors along which the optimum was varied/perturbed during exploration to obtain said extremapoint.

Since the singular vectors also serve as basis vectors in each m-dimensional space, the coefficients associated with the singular vectors (that indicate their magnitude of contribution) directly become the coordinates of the extremapoints. This way the extremapoint coordinates originally in d-dimensional space are reduced to a unique set of m-tuple coordinates (singular vector magnitudes) in m-dimensional space.



(a) (b)

Figure A2.1: In this figure $d = 3$ and $m = 2$. Only four extremapoints lie in the span of singular vectors 1 & 2. The coordinates in A2.1b indicate the magnitude/distance by which the optimum must be varied along the singular vector directions in order to reach the four extremapoints. The optimum (or exploration origin) is the green circle at coordinate (0,0).

A2.2 Common centres with multiple ellipsoids

Let $\boldsymbol{c}$ be the centre of the complete d-ellipsoid that is located at some distance from the optimum $\boldsymbol{f}$. Both these vectors are coordinates in the d-dimensional space with standard basis. Starting with an initial assumption, convex optimization iteratively updates the centre $\boldsymbol{c}$. While the centre vector $\boldsymbol{c}$ is defined in the standard basis, the m-polytope constraints are defined in the singular vector basis. Consequently, a change of basis is required

to convert the centre vector to the alternate basis. In the experiments, the inverse of the singular vector matrix (from the Hessian SVD at the optimum) is used as change of basis matrix.

This being known, a common centre vector $\boldsymbol{c}_{\text{sv}}$ can be defined in the d-dimensional space with singular vector basis as:

```
c_sv = inverseSVMatrix*(c-f);
```

Motivation: Exploration is always performed away from the optimum. Due to this, it is convenient to define the optimum as the origin for the d-dimensional space with singular vector basis. In relation to this origin/zero coordinate, the location of the centre of the d-ellipsoid is to be identified. This is equivalent to identifying the distance vector between the optimum and the d-ellipsoid's centre. The components of a distance vector contain the difference in coordinates between two points. The operation `(c-f)` reveals this distance vector in the standard basis. This vector is transformed into the singular vector basis space using `inverseSVMatrix`. This way the components of $\boldsymbol{c}_{\text{sv}}$ describe the coordinates of the centre relative to the origin (optimum) in terms of singular vector axes (instead of regular Cartesian axes).

Recall that each ellipsoid cross-section (m-ellipsoid) is defined in an m-dimensional space spanned by a subset of (m vectors out of) d total singular vectors. By this definition, the centre of each m-ellipsoid becomes the appropriate subset of the components of vector $\boldsymbol{c}_{\text{sv}}$.

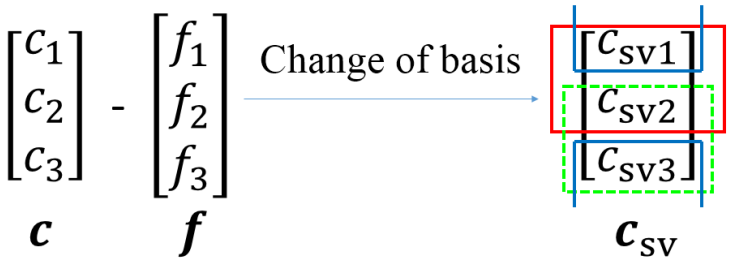


Figure A2.2: Refer to figure 4.5. With $d = 3$ & $m = 2$, ${}^3C_2 = 3$. In total three 2-ellipsoids are defined. The coloured boxes around the components of $\boldsymbol{c}_{\text{sv}}$ indicate the centres of each 2-ellipsoid. The optimum vector $\boldsymbol{f}$ is a constant while $\boldsymbol{c}$ is updated iteratively. As an example, $\boldsymbol{c}_{\text{sv1}}$ indicates the centre coordinate along the first singular vector.

An update to centre vector $\boldsymbol{c}$ over the course of optimization influences $\boldsymbol{c}_{\text{sv}}$. A change in one component of $\boldsymbol{c}_{\text{sv}}$ (the common centre vector) influences the centres of each m-ellipsoid that includes this component/singular vector axis.

Note: Treated separately, the ideal ellipsoid centre can be determined in each m-dimensional space. This allows for the largest m-dimensional ellipsoid to be inscribed within the m-polytope constraints for that space. To inscribe an ellipsoid in d-dimensions the m-dimensional ellipsoid cross-sections cannot be treated in isolation. The optimal centre vector strikes a compromise giving the largest inscribed ellipsoid in d-dimensions that still stays within each of the ${}^dC_m$ m-polytope constraints. In this context, it is acceptable if the individual cross-sections of the d-dimensional ellipsoid are not necessarily the largest.

Alternatively, the d-ellipsoid centre can be defined in advance. Conservatively, any coordinate in the interior of the convex hull of the extremapoints found during exploration can be selected. In the experiments, the optimum coordinate $\boldsymbol{f}$ is chosen as the centre. As a result, `c_sv = inverseSVMatrix*(f-f)` is a zero vector. This way, the centre of each m-ellipsoid is located at the origin of its m-space.

A2.3 Mathematical description of an ellipsoid

The derivation presented here is based on the text in [23] and [11].

A (closed) unit ball centered at the origin in d-dimensional space is represented by the set:

$$\{\boldsymbol{y} \in \mathbb{R}^d \mid \boldsymbol{y}^T\boldsymbol{y} \leqslant 1\} \tag{A2.1}$$

Let $\boldsymbol{B}$ be a $d \times d$ transformation matrix that is non-singular and positive definite ($\boldsymbol{B} \in \boldsymbol{S}^n_{++}$). An ellipsoid is an affine transformation of the unit ball. The ellipsoid ϵ centered at $\boldsymbol{c} \in \mathbb{R}^d$ can therefore be represented as:

$$\epsilon = \{\boldsymbol{B}\boldsymbol{y} + \boldsymbol{c} \mid \boldsymbol{y}^T\boldsymbol{y} \leqslant 1\} \tag{A2.2}$$

Changing the variable, if $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{y} + \boldsymbol{c}$ then $\boldsymbol{y} = \boldsymbol{B}^{-1}(\boldsymbol{x} - \boldsymbol{c})$

$$\epsilon = \{\boldsymbol{x} \mid (\boldsymbol{B}^{-1}(\boldsymbol{x} - \boldsymbol{c}))^T\boldsymbol{B}^{-1}(\boldsymbol{x} - \boldsymbol{c}) \leqslant 1\} \tag{A2.3}$$
$$\epsilon = \{\boldsymbol{x} \mid (\boldsymbol{x} - \boldsymbol{c})^T(\boldsymbol{B}^{-1})^T\boldsymbol{B}^{-1}(\boldsymbol{x} - \boldsymbol{c}) \leqslant 1\} \tag{A2.4}$$
$$\epsilon = \{\boldsymbol{x} \mid (\boldsymbol{x} - \boldsymbol{c})^T(\boldsymbol{B}\boldsymbol{B}^T)^{-1}(\boldsymbol{x} - \boldsymbol{c}) \leqslant 1\} \tag{A2.5}$$
$$\epsilon = \{\boldsymbol{x} \mid (\boldsymbol{x} - \boldsymbol{c})^T\boldsymbol{Q}^{-1}(\boldsymbol{x} - \boldsymbol{c}) \leqslant 1\} \tag{A2.6}$$

Where $\boldsymbol{Q} = \boldsymbol{B}\boldsymbol{B}^T$ and A2.6 is the usual expression for an ellipsoid in matrix form. The ellipsoid axes lengths and orientations can be extracted from $\boldsymbol{Q}$ by performing an SVD.

$$\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V} \tag{A2.7}$$

In equation A2.7, $\boldsymbol{Q}$ is expressed as the product of three matrices. Since $\boldsymbol{Q}$ is symmetric and positive definite, $\boldsymbol{V} = \boldsymbol{U}^T$. The columns of $\boldsymbol{U}$ known as singular vectors represent the axis orientations of the ellipsoid. Singular values are sorted in ascending order in the diagonal matrix $\boldsymbol{\Lambda}$. These relate to the semi-axis lengths as:

$$\text{semi-axis length}_i = \frac{1}{\sqrt{\text{singular value}_i}},\ i = 1, \ldots, d \tag{A2.8}$$

A2.4 Convex optimization formulation

The derivation presented here is based on the text in section 8.4.2 of [2]. The derivation assumes that both the polytope and inscribed ellipsoid are defined in a d-dimensional space.

Since the transformation matrix $\boldsymbol{B} \in \boldsymbol{S}^n_{++}$, (as mentioned in A2.3) the volume of an ellipsoid is proportional to the determinant of this matrix. The maximum volume ellipsoid can be found via the convex optimization problem:

$$\begin{aligned} &\text{maximize: } \log\det \boldsymbol{B} \\ &\text{subject to: } \sup_{\|\boldsymbol{y}\| \leqslant 1} I_{\boldsymbol{C}}(\boldsymbol{B}\boldsymbol{y} + \boldsymbol{c}) \leqslant 0 \end{aligned} \tag{A2.9}$$

Where $I_{\boldsymbol{C}}$ is the indicator function for a convex set $\boldsymbol{C}$

$$I_{\boldsymbol{C}}(\boldsymbol{x}) = \begin{cases} 0 & \text{for } \boldsymbol{x} \in \boldsymbol{C} \\ +\infty & \text{for } \boldsymbol{x} \notin \boldsymbol{C} \end{cases} \tag{A2.10}$$

sup is the supremum of vector y that restricts it to within a unit ball and at most unit magnitude.

The optimization formulation stated in A2.9 is applicable to a general case. This thesis specifically considers the problem of finding the maximum volume ellipsoid inscribed in a polytope where the polytope is represented by a set of linear inequalities:

$$\boldsymbol{C} = \{\boldsymbol{x} \mid \boldsymbol{a}_i\boldsymbol{x} \leqslant b_i,\ i = 1, \ldots, k\} \tag{A2.11}$$

k represents the number of inequalities. The polytope $\boldsymbol{C}$ consists of the set of all elements x for which these linear inequalities hold. Each inequality defines a unique facet of the polytope. Equation A2.11 is known as the $\boldsymbol{H}$-polytope or half-space polytope representation - for more details refer to Glossary 15.1 in [24] & Theorem 2.49 in [45]. When inscribing a maximum volume ellipsoid in a polytope, only the set of elements $\boldsymbol{x}$ that lie in the interior of both the ellipsoid and polytope are to be identified.

Mathematically, the set of elements in the interior of an ellipsoid is represented by A2.2. Knowing this, the constraint in A2.9 can be restructured to incorporate the polytope inequality from A2.11 in place of the indicator function $I_{\boldsymbol{C}}$ as:

$$\begin{aligned} &\sup_{\|\boldsymbol{y}\| \leqslant 1} I_{\boldsymbol{C}}(\boldsymbol{B}\boldsymbol{y} + \boldsymbol{c}) \leqslant 0 \\ \iff &\sup_{\|\boldsymbol{y}\| \leqslant 1} \boldsymbol{a}_i(\boldsymbol{B}\boldsymbol{y} + \boldsymbol{c}) \leqslant b_i,\ i = 1, \ldots, k \end{aligned} \tag{A2.12}$$

$$\iff \sup_{\|\boldsymbol{y}\| \leqslant 1} \boldsymbol{a}_i\boldsymbol{B}\boldsymbol{y} + \boldsymbol{a}_i\boldsymbol{c} \leqslant b_i,\ i = 1, \ldots, k \tag{A2.13}$$

As per the discussions in [31] the term $\boldsymbol{a}_i\boldsymbol{B}\boldsymbol{y}$ has maximum value when:

$$\boldsymbol{y} = \frac{(\boldsymbol{a}_i\boldsymbol{B})^T}{\|\boldsymbol{a}_i\boldsymbol{B}\|} \tag{A2.14}$$

This is because $\boldsymbol{y}$ has unit magnitude and is parallel to the vector $\boldsymbol{a}_i\boldsymbol{B}$. Therefore $\boldsymbol{y}$ can be replaced as:

$$\iff \frac{\boldsymbol{a}_i\boldsymbol{B}(\boldsymbol{a}_i\boldsymbol{B})^T}{\|\boldsymbol{a}_i\boldsymbol{B}\|} + \boldsymbol{a}_i\boldsymbol{c} \leqslant b_i,\ i = 1, \ldots, k \tag{A2.15}$$

Which finally reduces to:

$$\iff \left\|\boldsymbol{B}\boldsymbol{a}_i^T\right\| + \boldsymbol{a}_i\boldsymbol{c} \leqslant b_i,\ i = 1, \ldots, k \tag{A2.16}$$

Using A2.16, we can reformulate A2.9 as:

$$\begin{aligned} \textbf{Maximize}\ &\log\det \boldsymbol{B} \\ \textbf{Subject to}\ &\left\|\boldsymbol{B}\boldsymbol{a}_i^T\right\| + \boldsymbol{a}_i\boldsymbol{c} \leqslant b_i,\ i = 1, \ldots, k \end{aligned} \tag{A2.17}$$

Which is the regular optimization formulation for inscribing a d-dimensional ellipsoid in a d-polytope. The transformation matrix $\boldsymbol{B}$ and centre vector $\boldsymbol{c}$ are the optimization variables.

The main issue with this formulation is that the number of constraints scales exponentially in relation to the number of dimensions d.

To keep the problem realistic this thesis proposes a slightly different optimization formulation where the single d-polytope is replaced by ${}^dC_m$ lower dimensional m-polytopes. (m refers to the size of the subset of dimensions grouped together with $m <= d$) Some of the quirks relating to this formulation are presented below:

As mentioned in section 4.1, each m-polytope is defined in a unique m-dimensional space (m-space for short) where m orthonormal singular vectors replace the Cartesian axes as basis vectors. Each m-polytope constrains an m-dimensional cross-section of the complete d-dimensional ellipsoid. An m-dimensional cross-section is a unique m-dimensional ellipsoid (or m-ellipsoid for short) that is comprised of a subset of the axes of the complete d-ellipsoid.

Since the complete d-ellipsoid uses singular vector directions as axes orientations, each m-ellipsoid has axes orientation parallel to the singular vector basis of its m-space. See figure 4.5 for more details. The consequence of this definition is that the transformation matrix $\boldsymbol{B}$ in the alternate optimization formulation will be diagonal. Recall from appendix A2.3 that an ellipsoid is an affine transformation of the unit ball. A transformation matrix $\boldsymbol{B}$ defines the axis lengths and orientations of the ellipsoid. Any ellipsoid that has axis orientation parallel to the coordinate axes (basis vectors) will have $\boldsymbol{B}$ as a diagonal matrix since the initial unit ball is only subjected to an axis scaling transform and not rotation. The semi-axis lengths of such an ellipsoid are contained along the diagonal of $\boldsymbol{B}$.

Recall from appendix A2.2 that the centre vector for each m-ellipsoid is the subset of components of a common centre vector defined for the complete d-ellipsoid. On a similar note, the semi-axis lengths for each m-ellipsoid consist of the appropriate subset of the diagonal elements of matrix $\boldsymbol{B}$. The choice of d-ellipsoid axes included in each m-ellipsoid determines this subset of components.

Equation A2.8 expresses the relation between singular values and semi-axis lengths. With semi-axis lengths in $\boldsymbol{B}$ and axis orientations (from the SVD of the Hessian at the optimum) in $\boldsymbol{U}$, the matrix form $\boldsymbol{Q}$ of the d-ellipsoid inscribed within ${}^dC_m$ lower dimensional m-polytopes is obtained as:

$$\boldsymbol{Q} \coloneqq \boldsymbol{U}(\boldsymbol{B}\boldsymbol{B}^T)^{-1}\boldsymbol{U}^T \tag{A2.18}$$

To be clear, unlike equation A2.7 where $\boldsymbol{Q}$ is already known, equation A2.18 defines $\boldsymbol{Q}$ as the product of the matrices in the RHS. The operation $(\boldsymbol{B}\boldsymbol{B}^T)^{-1}$ converts the semi-axis length to singular values as shown in equation A2.8. Intuitively, this equation is equivalent to re-associating the ellipsoid semi-axis lengths (obtained from fitting lower dimensional cross-sections of the d-ellipsoid in ${}^dC_m$ m-polytopes) with axis orientations that are known beforehand. Determining $\boldsymbol{Q}$ is essential since the ellipsoid matrix form serves as the basis for random sampling in the validation stage.

As seen with equation A2.11, linear inequalities define the facets of a polytope and are supplied as constraints to the optimization formulation. With ${}^dC_m$ different m-polytopes, the linear inequalities that define the facets of each m-polytope need to be accounted for separately.

Tying all this together, the final optimization formulation becomes:

$$
\begin{aligned}
&\textbf{Maximize} \quad \log\det \boldsymbol{B} \\
&\textbf{Subject to} \\
&\qquad \boldsymbol{c}_{\text{sv}} = \text{inverseSVMatrix} \times (\boldsymbol{c} - \boldsymbol{f}) \\
&\qquad \textbf{for } j \in \{1, \ldots, {}^dC_2\} \\
&\qquad\quad k^{\max} \leftarrow k^j \\
&\qquad\quad \boldsymbol{A} \leftarrow \boldsymbol{A}^j \\
&\qquad\quad \boldsymbol{b} \leftarrow \boldsymbol{b}^j \\
&\qquad\quad \text{(SV1,SV2)} \leftarrow \text{(2-plane)}^j \\
&\qquad\quad \boldsymbol{B}^j \leftarrow \begin{bmatrix} \boldsymbol{B}(\text{SV1,SV1}) & 0 \\ 0 & \boldsymbol{B}(\text{SV2,SV2}) \end{bmatrix} \\
&\qquad\quad \boldsymbol{c}^j_{\text{sv}} \leftarrow \begin{bmatrix} \boldsymbol{c}_{\text{sv}}(\text{SV1}) \\ \boldsymbol{c}_{\text{sv}}(\text{SV2}) \end{bmatrix} \\
&\qquad\quad\quad \textbf{for } i \in \{1, \ldots, k^{\max}\} \\
&\qquad\quad\quad\quad \boldsymbol{a}_i \leftarrow \boldsymbol{A}(\boldsymbol{i}, \boldsymbol{:}) \\
&\qquad\quad\quad\quad \left\| \boldsymbol{B}^j \boldsymbol{a}_i^T \right\| + \text{scaling} + \boldsymbol{a}_i \boldsymbol{c}^j_{\text{sv}} \leqslant b_i \\
&\qquad\quad\quad \text{end } \textbf{for} \\
&\qquad \text{end } \textbf{for}
\end{aligned}
\tag{A2.19}
$$

The above formulation is defined specifically for the case where $m = 2$. This formulation has been implemented and tested in MATLAB.

$\boldsymbol{c}_{\text{sv}}$ is the common centre vector defined with singular vector basis as mentioned in appendix A2.2. Index variable j has a maximum value of ${}^dC_2$ and represents the number of unique two dimensional planes that can be generated as pairwise combinations of the d total singular vectors.

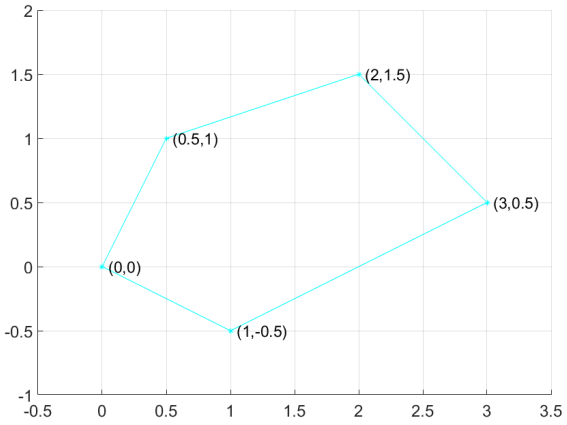
The value of $k^{\max}$ depends on the total number of facet defining inequality constraints specified for the j^{th} 2-plane. Each row of the matrix $\boldsymbol{A}$ ($k^{\max} \times 2$, where 2 is the constraint polytope dimension) and the corresponding element of vector $\boldsymbol{b}$ ($k^{\max} \times 1$) define one facet of the j^{th} 2-polytope. It is assumed that $\boldsymbol{A}$ and $\boldsymbol{b}$ have been calculated beforehand for each 2-plane. The operation (SV1,SV2) $\leftarrow$ (2-plane)^j extracts the indices of the two singular vectors that define the j^{th} 2-plane. These singular vector indices are used to determine:

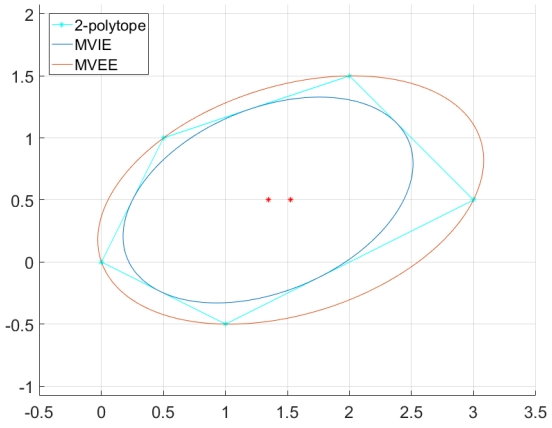
1. $\boldsymbol{B}^j$ - the subset of diagonal elements of the transformation matrix $\boldsymbol{B}$
2. $\boldsymbol{c}^j_{\text{sv}}$ - the subset of components of the common centre vector $\boldsymbol{c}_{\text{sv}}$

Both $\boldsymbol{B}^j$ (semi-axis lengths) and $\boldsymbol{c}^j_{\text{sv}}$ (centre vector) are only applicable to the relevant j^{th} 2-ellipsoid. Lastly a modified version of equation A2.17 is used (in the nested **for**) to test the fit of this j^{th} 2-ellipsoid within its 2-polytope constraint. A scaling constant is introduced that can be set as a small non-negative value to ensure a more conservative ellipsoid fit. (The effect on 2-ellipsoid size is seen in figure A2.4)

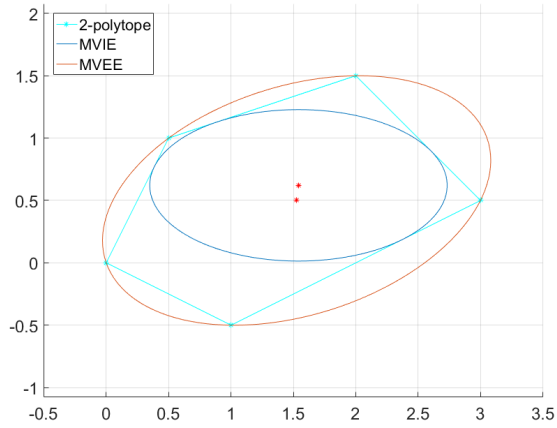
A2.5 Playing with CVX

The CVX optimization package [19][20] is used for fitting the maximum volume inscribed ellipsoid (MVIE) in a 2-polytope. Some results [1] are presented below where the effects of a pre-determined centre, diagonal transformation matrix $\boldsymbol{B}$ and scaling can be observed on the MVIE axes lengths and orientation. (Note: equation A2.17 is used here as the problem consists of inscribing a 2-ellipsoid in a 2-polytope) For comparison, a minimum volume enclosing ellipsoid (MVEE) generated with the Khachiyan algorithm (Khachiyan (1996) [28]) is displayed. The MATLAB implementation of this algorithm sourced from Moshtagh (2006) [34] is used.

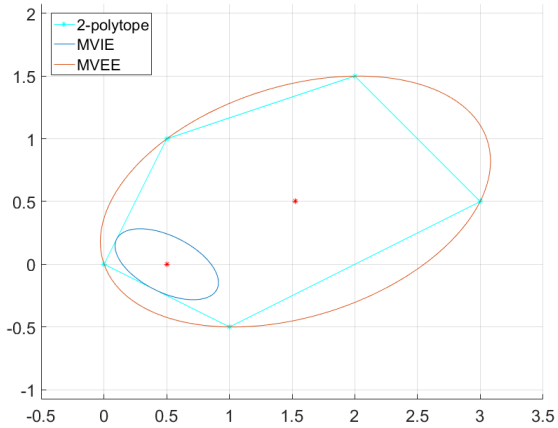




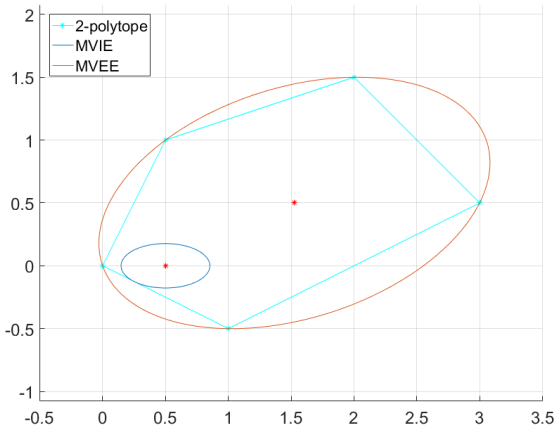
(a) Optimizer decides axes lengths and orientation - Largest ellipsoid possible



(b) Diagonal transformation matrix $\boldsymbol{B}$ is used - Largest axis aligned ellipsoid



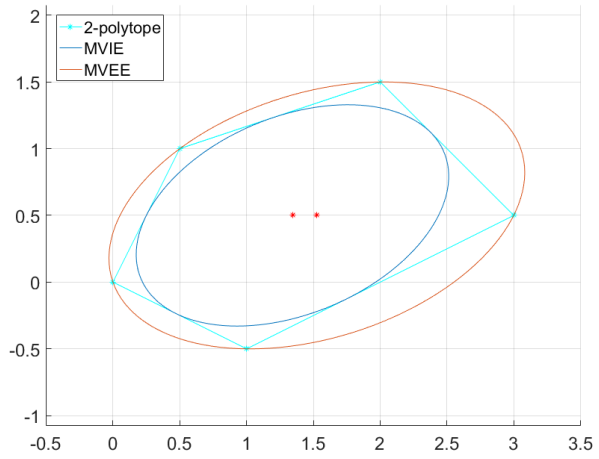
(c) Optimizer decides axes lengths and orientation - Largest ellipsoid possible with centre fixed at (0.5,0)



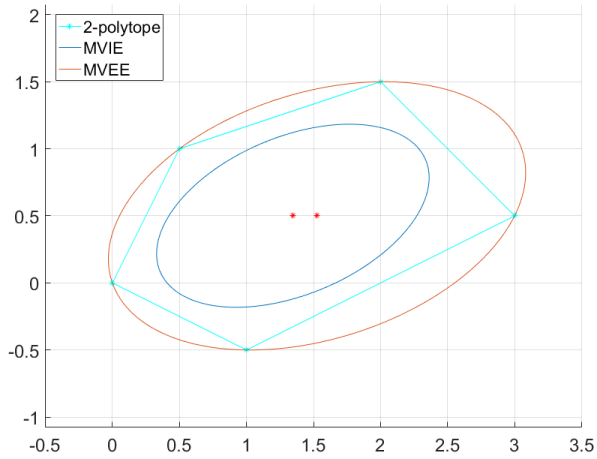
(d) Diagonal transformation matrix $\boldsymbol{B}$ is used - Largest axis aligned ellipsoid possible with centre fixed at (0.5,0)

Figure A2.3: Fit comparison: MVIE vs. MVEE

[1] The MATLAB source code for this example is provided here: http://web.cvxr.com/cvx/examples/cvxbook/Ch08_geometric_probs/html/max_vol_ellip_in_polyhedra.html



(a) Optimizer decides axes lengths and orientation - Largest ellipsoid possible



(b) Optimizer decides axes lengths and orientation - Largest ellipsoid possible with scaling coefficient of 0.15

Figure A2.4: Introducing a scaling coefficient as seen in the nested **for** in A2.19 into A2.17 allows for a more conservative ellipsoid fit

Since the MVEE algorithm encloses all vertices of the 2-polytope, the resultant 2-ellipsoid's centre, axis lengths and orientations are uniquely defined by these vertices. In the d-dimensional reservoir case, this implies an MVEE can include non-feasible regions since only $2 \times d$ vertices (extremapoints at the boundary of the feasible region) are responsible for defining the ellipsoid.

A2.6 Validation: Sample generation on ellipsoid surface

For this approach the uniform distribution of points generated in the ellipsoid interior using the workflow and code mentioned in [8] are projected to the ellipsoid surface. Recall the ellipsoid matrix equation:

$$(\boldsymbol{x} - \boldsymbol{c})^T \boldsymbol{Q}^{-1} (\boldsymbol{x} - \boldsymbol{c}) = 1 \tag{A2.20}$$

The inequality has been replaced by an 'equal to' sign as all points (including $\boldsymbol{x}$) should exist on the ellipsoid surface.

Let $\boldsymbol{y}$ be a point in the ellipsoid interior, located at some distance from the centre $\boldsymbol{c}$. By treating $\boldsymbol{y}$ as a direction vector and $\boldsymbol{c}$ as the origin, a point can be obtained on the ellipsoid surface as long as the magnitude of the direction vector is known. Let g be this magnitude. In that case, the point on the surface is $(\boldsymbol{c} + g\boldsymbol{y})$. Substituting this term in equation A2.20 in place of $\boldsymbol{x}$ we have:

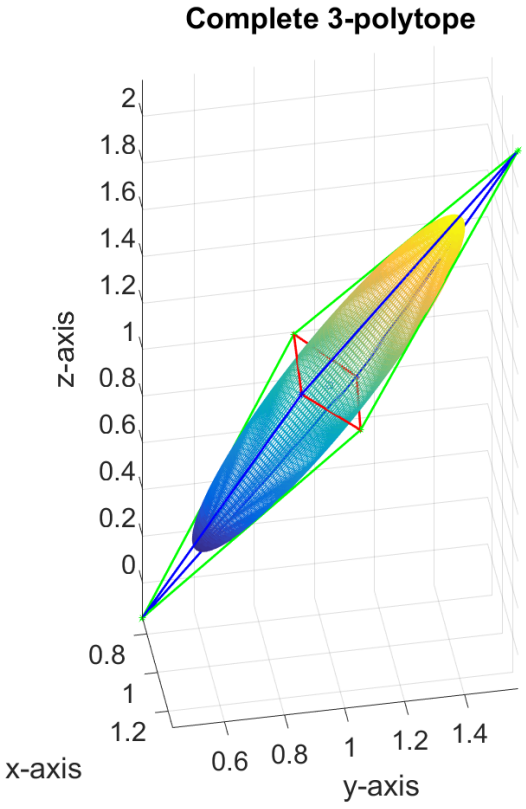
$$((\boldsymbol{c} + g\boldsymbol{y}) - \boldsymbol{c})^T \boldsymbol{Q}^{-1} ((\boldsymbol{c} + g\boldsymbol{y}) - \boldsymbol{c}) = 1 \tag{A2.21}$$

Which reduces to:

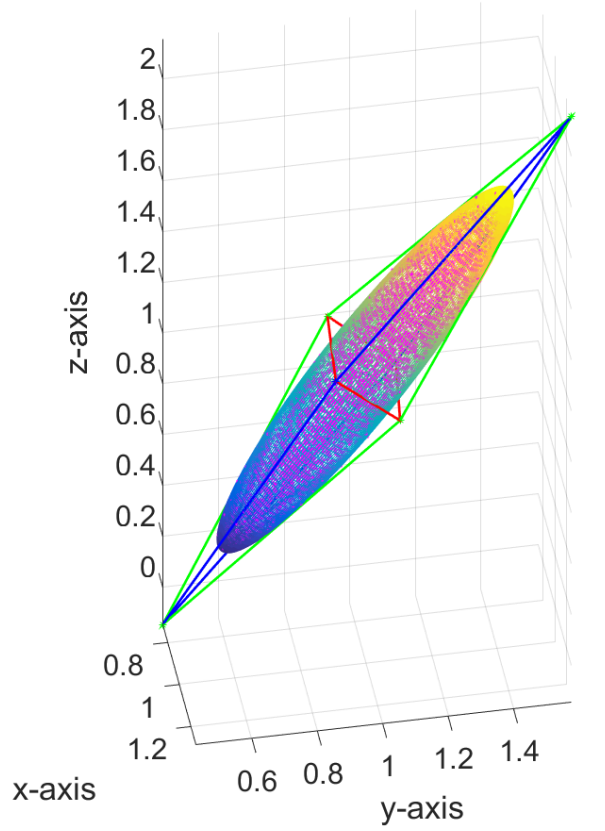
$$\begin{aligned} g^2 \boldsymbol{y}^T \boldsymbol{Q}^{-1} \boldsymbol{y} &= 1 \\ g &= \sqrt{\frac{1}{\boldsymbol{y}^T \boldsymbol{Q}^{-1} \boldsymbol{y}}} \end{aligned} \tag{A2.22}$$

As the ellipsoid is symmetric, a second point is obtained on the opposite side also on the ellipsoid surface as $(\boldsymbol{c} - g\boldsymbol{y})$.

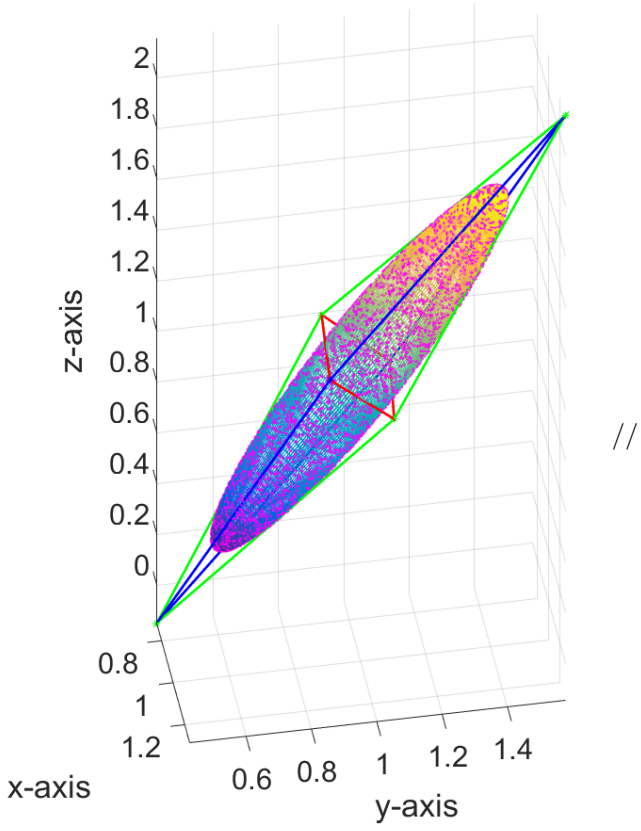
To be clear, the workflow mentioned in [8] gives a uniform distribution of points in the interior of the ellipsoid. It is yet to be determined if the nature of the distribution of points on the ellipsoid surface is also uniform.



(a)

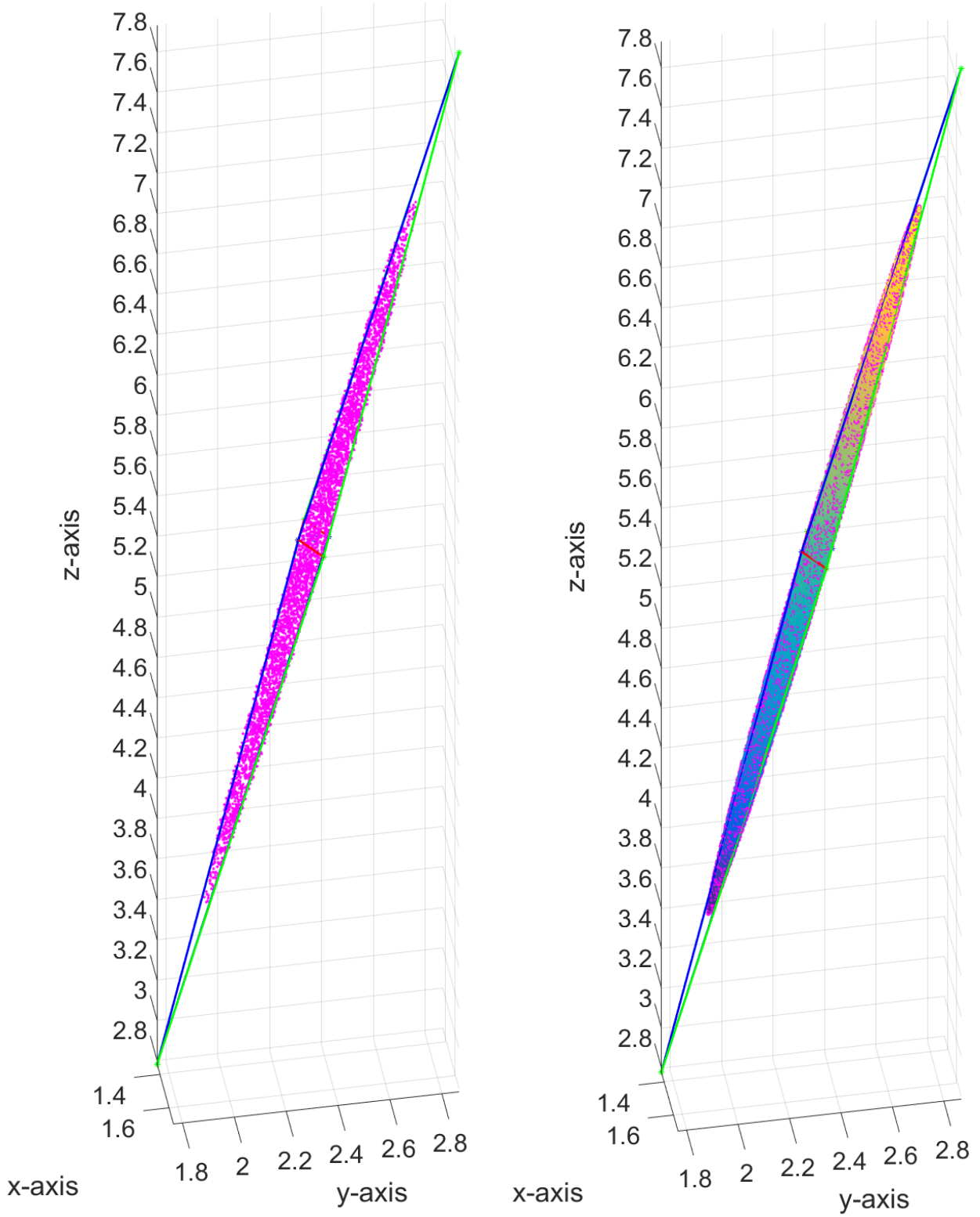


(b) Uniform distribution of points in the ellipsoid interior



(c) Points projected on ellipsoid surface

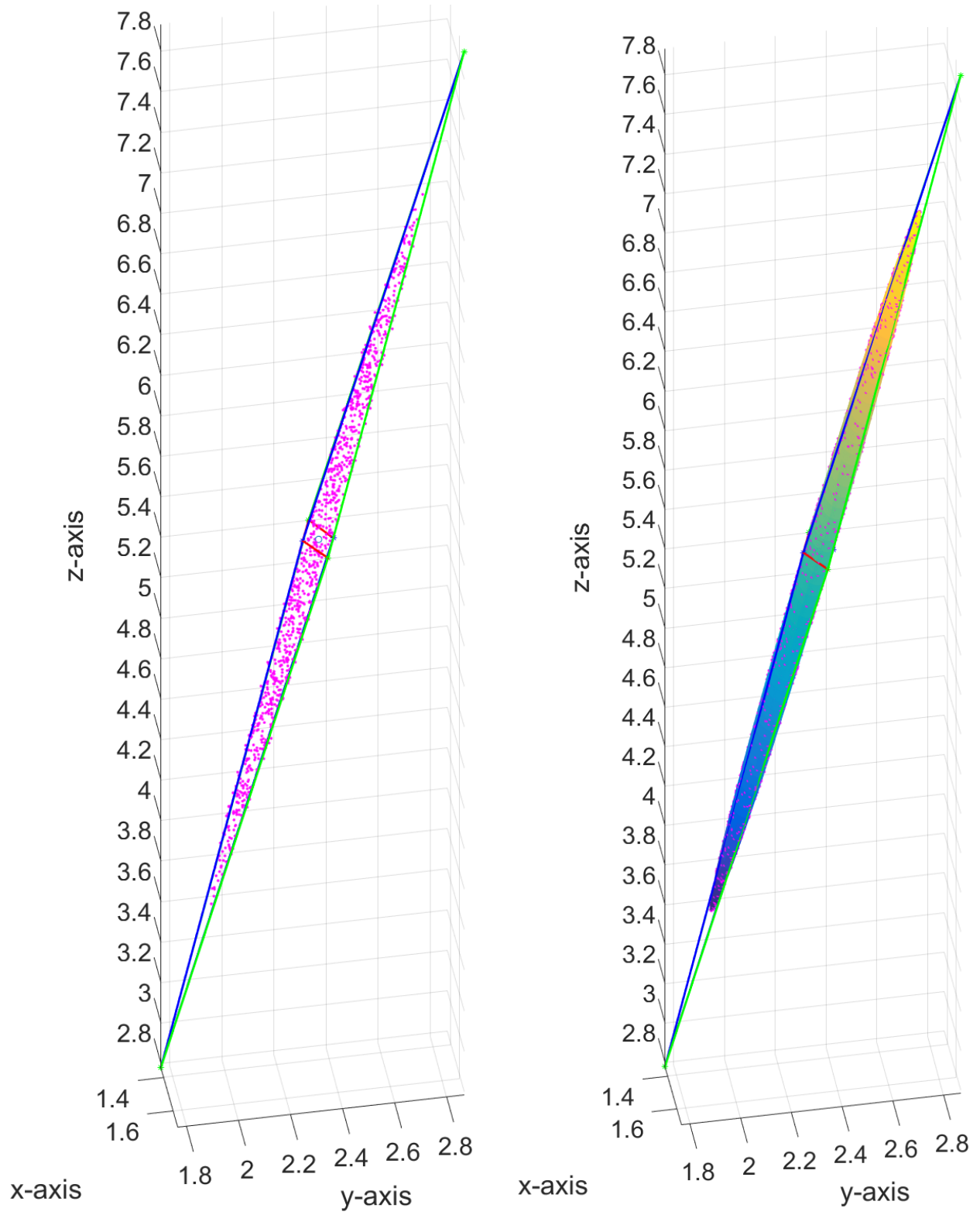
Figure A2.5: **Testing on the Rosenbrock function:** Exploration was conducted from [1,1,1] - the true optimum for the 3D Rosenbrock function. Exploration was terminated when a function value of 12 was reached. The ellipsoid was inscribed using the algorithm presented in A2.19. (4000 points/ellipsoid)



(a) Uniform distribution of points in the ellipsoid interior (ellipsoid not shown for better point visibility)

(b) Points projected on ellipsoid surface

Figure A2.6: Similar to the previous image A2.5, exploration was conducted from [1.526,2.338,5.461] - an origin chosen solely because of the resulting elongated ellipsoid. Exploration was terminated when a function value of 12 was reached. The uniformly distributed points in the ellipsoid interior are projected to the surface. Despite the approximately 40:1 ratio between the longest and shortest axes lengths, the distribution of points in the ellipsoid interior and on the ellipsoid surface remains visually uniform. (sampling density: 4000 points/ellipsoid)



(a) Uniform distribution of points in the ellipsoid interior (ellipsoid not shown for better point visibility)

(b) Points projected on ellipsoid surface

Figure A2.7: Similar to the previous image A2.6, the distribution of points in the ellipsoid interior and on the ellipsoid surface remains visually uniform even with a reduced sampling density of 1000 points/ellipsoid.

Bibliography

[1] Baker, C.A., Watson, L.T., Grossman,B., Mason, W.H., Cox,S.E., Haftka, R.T. Study of a Global Design Space Exploration Method for Aerospace Vehicles http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.3185&rep=rep1&type=pdf

[2] Boyd, Stephen & Vandenberghe, Lieven. (2004) Convex Optimization. Cambridge University Press https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

[3] Brouwer, D.R., Jansen, J.D. (2004) Dynamic Optimization of Waterflooding With Smart Wells Using Optimal Control Theory. SPE Journal https://www.onepetro.org/journal-paper/SPE-78278-PA

[4] Chen, Wai-Kai (2009) Computer Aided Design and Design Automation. The Circuits and Filters Handbook, Third Edition. https://www.crcpress.com/Computer-Aided-Design-and-Design-Automation/Chen/p/book/9781420059182

[5] Chen, Y. (2008) Efficient ensemble based reservoir management. PHD thesis. University of Oklahoma

[6] Chen, Y., Oliver, D.S., Zhang, D. (2009) Efficient Ensemble-Based Closed-Loop Production Optimization. SPE Symposium on Improved Oil Recovery, 20-23 April, Tulsa, Oklahoma, USA. https://www.onepetro.org/journal-paper/SPE-112873-PA

[7] Dennis, John E., & Schnabel Robert B. (1983) Numerical Methods for Unconstrained Optimization and Nonlinear Equations. http://epubs.siam.org/doi/book/10.1137/1.9781611971200

[8] Dezert, Jean & Musso, Christian. ONERA/DTIM, France. An Efficient Method for Generating Points Uniformly Distributed in Hyperellipsoids https://www.onera.fr/sites/default/files/297/C013_-_Dezert_-_YBSTributeMonterey2001.pdf

[9] Director, S., Hachtel, G. (1977) The simplicial approximation approach to design centering. http://ieeexplore.ieee.org/document/1084353/

[10] Eyerman, S., Eeckhout, L., Bosschere, K.D. (2006) Efficient Design Space Exploration of High Performance Embedded Out-of-Order Processors http://ieeexplore.ieee.org/document/1656905/

[11] Feige, Uri & Krauthgamer, Robi. (2008) Advanced Algorithms - Handout 7: The Ellipsoid algorithm and Positive Definite matrices. http://www.wisdom.weizmann.ac.il/~robi/teaching/AdvAlgs2008/handout7.pdf

[12] Fonseca, R.M. (2011) Robust ensemble based multi-objective production optimization: Application to smart mells. MSc thesis. https://repository.tudelft.nl/islandora/object/uuid%3Abdda7a33-1073-4a38-aabb-124367b3a3e1

[13] Fonseca, R.M., Kahrobaei, S.S., Van Gastel, L.J.T., Leeuwenburgh, O., Jansen, J.D. (2015a) Quantification of the Impact of Ensemble Size on the Quality of an Ensemble Gradient Using Principles of Hypothesis Testing. SPE Reservoir Simulation Symposium, 23-25 February, Houston, Texas, USA https://www.onepetro.org/conference-paper/SPE-173236-MS

[14] Fonseca, R.M., Leeuwenburgh, O., Rossa, E.D., Van den Hof, P.M.J., Jansen, J.D. (2015b) Ensemble-Based Multiobjective Optimization of On/Off Control Devices Under Geological Uncertainty. SPE Reservoir Evaluation & Engineering https://www.onepetro.org/journal-paper/SPE-173268-PA

[15] Fonseca, R.M., Chen B., Jansen, J.D., Reynold, Albert. (2017) A Stochastic Simplex Approximate Gradient (StoSAG) for optimization under uncertainty. International Journal for Numerical Methods in Engineering Published by John Wiley & Sons, Ltd. http://onlinelibrary.wiley.com/doi/10.1002/nme.5342/full

[16] Galton, A., Duckham, M. (2006) What Is the Region Occupied by a Set of Points? http://www.geosensor.net/papers/galton06.GISCIENCE.pdf

[17] Gao, G., Reynolds, A.C. (2006) An Improved Implementation of the LBFGS Algorithm for Automatic History Matching. SPE Journal. https://www.onepetro.org/journal-paper/SPE-90058-PA

[18] Gill, R.P., Leonard M.W. (2001) Reduced-Hessian quasi-Newton methods for unconstrained optimization. Society for Industrial and Applied Mathematics. http://www.ccom.ucsd.edu/~peg/papers/rhr_siam.pdf

[19] Grant, Michael & Boyd, Stephen. (2008) Graph implementations for nonsmooth convex programs, Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, pages 95-110, Lecture Notes in Control and Information Sciences, Springer. http://stanford.edu/~boyd/graph_dcp.html

[20] Grant, Michael & Boyd, Stephen. (2013) CVX: Matlab software for disciplined convex programming, version 2.0 beta. http://cvxr.com/cvx

[21] Gries, Matthias. (2003) Methods for Evaluating and Covering the Design Space during Early Design Development https://www.sciencedirect.com/science/article/pii/S016792600400032X

[22] Griffin, Christopher. (2012) Numerical Optimization: Penn State Math 555 Lecture Notes http://www.personal.psu.edu/cxg286/Math555.pdf

[23] Gupta, Anupam & O'Donnell, Ryan. (2011) 15-859(E) Linear and Semidefinite Programming (Advanced Algorithms) - Lecture 8: The Ellipsoid Algorithm for LP feasibility. https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15859-f11/www/notes/lecture08.pdf

[24] Henk, M., Richter-Gebert, Jurgen & Ziegler, Gunter M. (1997) Basic properties of convex polytopes. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.136.2061&rep=rep1&type=pdf

[25] Jansen, J.D., Brouwer, D.R., Sippe G. Douma. (2009) Closed loop reservoir management. SPE Journal https://www.onepetro.org/conference-paper/SPE-119098-MS

[26] Kang, S., Kumar, R. Magellan: A Framework for Fast Multi-core Design Space Exploration and Optimization Using Search and Machine Learning https://pdfs.semanticscholar.org/24f3/032e84b307ceafbefd4d81be9c19051b91a2.pdf

[27] Karl, W.C., Verghese, G.C., Willsky, A.S. (1994) Reconstructing Ellipsoids from Projections. CVGIP: Graphical Models and Image Processing. Volume 56, Issue 2, Pages 124-139 https://www.sciencedirect.com/science/article/pii/S1049965284710121

[28] Khachiyan, L.G. (1996) Rounding of polytopes in the real number model of computation. Mathematics of Operations Research 21, 307–320. https://doi.org/10.1287/moor.21.2.307

[29] Li, Guipeng. Code for strong Wolfe line search and zoom. GitHub repository. https://github.com/GuipengLi/optLBFGS/blob/master/optLBFGS.m

[30] Lie, K.-A. (2016) An Introduction to Reservoir Simulation Using MATLAB: User guide for the Matlab Reservoir Simulation Toolbox (MRST). SINTEF ICT. https://www.sintef.no/contentassets/8af8db2e42614f7fb94fb0c68f5bc256/mrst-book-2016.pdf

[31] Löfberg, Johan (2017). Finding the maximum volume inscribed ellipsoid using CVX. Communication on stackexchange. https://math.stackexchange.com/questions/2456854/finding-the-maximum-volume-inscribed-ellipsoid-using-cvx

[32] Matt J. Analyze N-dimensional Polyhedra in terms of Vertices or (In)Equalities. https://nl.mathworks.com/matlabcentral/fileexchange/30892-analyze-n-dimensional-polyhedra

[33] Moreira, A., Santos, M.Y. (2007) Concave hull: a k-nearest neighbours approach for the computation of the region occupied by a set of points https://repositorium.sdum.uminho.pt/bitstream/1822/6429/1/ConcaveHull_ACM_MYS.pdf

[34] Moshtagh, N. (2006) Minimum volume enclosing ellipsoids http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.7691&rep=rep1&type=pdf MATLAB code: https://nl.mathworks.com/matlabcentral/fileexchange/9542-minimum-volume-enclosing-ellipsoid

[35] Moshtagh, N. (2007) Plot an ellipse in "center form" MATLAB code: https://nl.mathworks.com/matlabcentral/fileexchange/13844-plot-an-ellipse-in--center-form-

[36] Nocedal, Jorge; Wright, Stephen J. (2006) Numerical Optimization. Springer

[37] Oliver, D.S., Reynolds, A.C., Liu N. (2008) Numerical Optimization. Inverse Theory for Petroleum Reservoir Characterization and History Matching. Cambridge University Press. https://doi.org/10.1017/CBO9780511535642

[38] Strang, Gilbert. (2006) Linear Algebra and its application. Fourth edition. Thomson Brooks/Cole

[39] Van Essen, G., Zandvliet, M., Van den Hof, P.M.J., Okko Bosgra, Jansen, J.D. (2009) Robust Waterflooding Optimization of Multiple Geological Scenarios. SPE Journal https://www.onepetro.org/journal-paper/SPE-102913-PA

[40] Van Essen, G., Van den Hof, P.M.J., Jansen, J.D. (2011) Hierarchical Long-Term and Short-Term Production Optimization. SPE Journal https://www.onepetro.org/journal-paper/SPE-124332-PA

[41] Van Essen, G., Van den Hof, P.M.J., Jansen, J.D. (2012) A Two-Level Strategy to Realize Life-Cycle Production Optimization in an Operational Setting. SPE Intelligent Energy International, 27-29 March, Utrecht, The Netherlands. https://www.onepetro.org/conference-paper/SPE-149736-MS

[42] Wang, C., Li, G., Reynolds, A.C. (2007) Production Optimization in Closed-Loop Reservoir Management. SPE Annual Technical Conference and Exhibition, 11-14 November, Anaheim, California, U.S.A. https://www.onepetro.org/conference-paper/SPE-109805-MS

[43] Williamson, David P. (2014). ORIE 6300 Mathematical Programming I - Lecture 4: class notes. https://people.orie.cornell.edu/dpw/orie6300/Lectures/lec04.pdf

[44] Zamora-Sillero, E., Hafner, M., Ibig, A., Stelling, J., Wagner, A. (2011) Efficient characterization of high-dimensional parameter spaces for systems biology https://bmcsystbiol.biomedcentral.com/articles/10.1186/1752-0509-5-142

[45] Ziegler, Gunter M. (2013) Discrete geometry I. https://wikis.fu-berlin.de/download/attachments/528515139/Ziegler+DiscGeo+Lec+Notes+Part+1.pdf