Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Jakob Eide Grepperud

Stochastic Gradient Optimization of Petroleum Assets

Towards Reinforcement Learning

Master's thesis in Cybernetics and Robotics
Supervisor: Bjarne Grimstad
Co-supervisor: Lars Imsland
June 2022

Jakob Eide Grepperud

Stochastic Gradient Optimization of Petroleum Assets

Towards Reinforcement Learning

Master’s thesis in Cybernetics and Robotics
Supervisor: Bjarne Grimstad
Co-supervisor: Lars Imsland
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Stochastic Gradient Optimization of Petroleum Assets: Towards Reinforcement Learning

Jakob Eide Grepperud

June 20, 2022

Preface

This thesis was written in the final semester of the Master's programme of Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU) in Trondheim. It constitutes the completion of my studies at the Department of Engineering Cybernetics. The thesis is a continuation of the project report from the fall of 2021, and is written in collaboration with Solution Seeker.

The motivation for the topic of this thesis was to further build on the work done on stochastic optimization from this fall. As reinforcement learning falls within this general term, I was very motivated to explore this class of methods because of their potential. I am very happy that I got a relatively creative task, allowing me to properly understand the reinforcement learning framework which I earlier perceived as magic.

I would like to thank my supervisors, Bjarne Grimstad and Lars Imsland, for their guidance and insight. I would especially like to thank Bjarne for formulating the project and entrusting me with it. Thank you for the very necessary meetings and for helping me not get lost in the forest of possible approaches.

Finally, a big thank you to my family for their continuing support.

Jakob Eide Grepperud

Trondheim, 20th June 2022

Abstract

A petroleum asset presents many unique challenges for real-time optimization. Developing a process model based on first-principles is complicated and expensive due to highly complex dynamics and a lack of available instrumentation. Furthermore, the extraction process induces time-varying dynamics, requiring continuous calibration of the process model to remain accurate. Model-free optimization methods have been receiving increased attention because of this. However, as many of these methods fail to consider the various limitations of a production asset, operator engineers still decide on most control changes.

In this thesis, we investigate the use of policy gradient methods for gas-lifted petroleum optimization. These methods form the intersection between stochastic gradient methods and reinforcement learning – a class of machine learning whose popularity has grown exponentially over recent years. Our goal was to evaluate the applicability of policy gradient methods to real-time petroleum optimization. To the extent of our knowledge, there is yet no research on policy gradient methods for the gas-lift problem.

Three different policy gradient algorithms, based on single-sample Monte Carlo estimators, are tested and compared using Stochastic gradient ascent. The SPSA algorithm is used as a benchmark, employing a stochastic finite difference gradient from two samples. Several synthetic test cases are implemented to evaluate robustness against various phenomena, such as measurement noise, time-varying optimum, and system-wide constraints.

The results show that the Actor-critic algorithm performs the best on systems with fewer wells, using a linear approximator for the value function. The SPSA algorithm performs better with higher dimensions. The policy gradient methods are generally more robust toward new scenarios than SPSA, with natural gradients accelerating the optimization process. All algorithms converge to optimal production within a few iterations in most simulations. Although they perform well on the synthetic test cases, several research areas should be explored before their applicability to real-time optimization can be concluded. However, we believe these methods might have the potential to be a valuable tool for operators in the future.

Sammendrag

Et petroleumsproduksjonssystem byr på flere unike utfordringer som gjør sanntidsoptimering vanskelig. Modellutvikling basert på fysiske lover er komplisert og kostbart på grunn av svært kompleks dynamikk og mangel på tilgjengelig sensorikk. Ekstraksjonsprosessen fører til tidsvarierende dynamikk som gjør at enhver prosessmodell må kontinuerlig kalibreres for å opprettholde nødvendig nøyaktighet. Grunnet dette er det stadig ny forskning på modellfrie optimeringsmetoder. Til tross for ny forskning utføres de aller fleste kontrollbeslutningene av menneskelige operatører. Dette er fordi metodene sjelden tar hensyn til begrensningene som oppstår ved optimering av produksjonssystem.

I denne oppgaven undersøker vi bruk av policy gradient-metoder for gassløftede produksjonssystemer. Disse metodene utgjør fellesnevneren til stokastisk gradientsøk og forsterkende læring – en klasse av maskinlæringsmetoder som har fått eksponensielt økende oppmerksomhet de siste årene. Vårt hovedmål var å evaluere potensialet for at policy gradient-metoder kan passe til sanntidsoptimering av petroleumssystemer. Etter vår beste evne har vi ikke funnet noen tidligere forskning på anvendelser på gassløft-problemet.

Tre forskjellige gradient-metoder ble implementert og testet ved bruk av en Stochastic gradient ascent-algoritme. Alle baserer seg på Monte Carlo-estimatorer som bruker kun én måling. SPSA-algoritmen er brukt som et referansepunkt, siden den baserer seg på finite difference-estimering, en litt annerledes tilnærming. Flere syntetiske testmodeller er konstruert for å simulere ulike fenomener som kan skje under gassløft, inkludert støy, tidsvarierende optimum og beskrankninger. Dette er for å evaluere robusthet.

Resultatene viser at SPSA-algoritmen yter bedre med flere antall brønner. Actor-critic algoritmen yter best med færre brønner. Dette er en algoritme som benytter seg av en verdifunksjon som læres underveis i simuleringen. En lineær funksjonsapproksimator brukes for å modellere denne. Policy gradient-metodene er generelt mer robust mot ulike scenarior enn SPSA. Vi har brukt naturlige gradienter til å akselerere optimeringsprosessen. Algoritmene yter bra på testmodellene. Det kreves likevel ytterlig forskning på flere områder, for å kunne konkludere om disse metodene egner seg for sanntidsoptimering. Likevel tror vi at disse metodene kan ha potensiale til å bli brukt som et assisterende verktøy for operatører i fremtiden.

Contents

Figures

Tables

Acronyms

A2C Advantage Actor-Critic. 18

AC Actor-Critic. 17–19, 21, 33–36, 41, 43–45, 59, 64

ACKTR Actor-Critic using Kronecker-Factored Trust Region. 19

ANN artificial neural network. 8, 16, 28, 45, 46

CAPG Clipped Action Policy Gradient. vi, 19, 20, 26, 27, 34, 43

DP dynamic programming. 8, 15

DPG Deterministic Policy Gradient. 15

ESC extremum seeking control. 7

FD finite difference. 12, 13, 16

GAE General Advantage Estimation. 18

KL Kullback-Leibler. 18, 19

LLS linear least squares. 28

MAE Mean Absolute Error. vi, 32, 34–38, 40, 41

MDP Markov decision process. 13–15, 17

MVP mean-value problem. vi, 10, 11

PG policy gradient. 16–19, 43, 45, 46

PPO Proximal Policy Optimization. 19, 20

RL reinforcement learning. 3, 4, 8–10, 13–16, 18, 19, 26, 43–45, 47

RMSE Root Mean Squared Error. 32, 34–38, 40–42, 45

RTO real-time optimization. 1, 2, 6, 16, 33

S1 Score1 with Clipped Action Policy Gradient. 34, 35, 40, 41, 43, 58

Score1 Single-Sample Score Function Gradient. 18, 27, 31, 33, 34, 40

SD standard deviation. 24, 26, 34, 43, 66–68

SGD Stochastic gradient descent. 11, 17, 21, 25, 26, 30

SNG Score1 with Natural Gradient. 34–45, 58, 63–66

SPSA Simultaneous Perturbation Stochastic Approximation. 7, 12, 13, 16, 21, 29–31, 33–45, 47, 59

TD temporal-difference. 17, 18, 27, 28, 42

TRPO Trust-Region Policy Optimization. 19

Chapter 1

Introduction

Over the last decade, an emerging field of computational intelligence has shown groundbreaking results, giving rise to new speculations about the thus far seemingly unachievable concept of general artificial intelligence (Silver, Singh et al., 2021). One famous example of such a breakthrough is AlphaGo, the first computer program to beat a human professional in the ancient game of Go, widely considered one of the most demanding challenges of artificial intelligence (Mandziuk, 2007; Silver, Huang et al., 2016). With a state-space short of 3^{361} combinations and complex tactics needed on different time scales, methods used to master other games like chess fail to beat even intermediate players. This feat was possible not by teaching the algorithm how to play but by letting it become its own teacher, essentially only giving it the rules of the game. Headlines like these have received attention across all areas of engineering, and the process industry is not an exception.

As the world population increases along with the global standard of living, the electricity demand continues to outgrow the renewable energy supply. At the same time, the oil industry might be facing a future with fewer production assets, with new reports calling for a finite end horizon to the search and extraction of fossil fuels (HDR, 2021; IEA, 2021a,b). Thus, there is a need for innovative solutions that fully utilize existing assets to meet a future with volatile prices, tighter profit margins, and competitive markets. One such solution may be new approaches to *daily production optimization*, where control allocation and other decisions are made to maximize production efficiency. Planning and optimizing a petroleum system is complex and might happen on a time scale of years. Daily optimization happens over a time scale of hours to days, making it equivalent to *real-time optimization* (RTO) from a process perspective (Foss et al., 2018). Amongst having the potential to reduce power consumption, daily optimization has been shown to increase production volume by 1 to 7%, as well as assist operators in reducing workload (Krishnamoorthy et al., 2019; Palen and Goodwin, 1996). However, RTO in the petroleum industry has yet to see widespread usage, despite being a research topic of high-interest (Bieker et al., 2006). A petroleum asset presents a unique set of challenges that must be addressed by any optimization method. Krishnamoorthy et al. (2019) list two of these: offline model development and online model adjustments.

Traditional, model-based methods need accurate modeling to successfully solve numerical optimization problems that occur during the lifetime of the plant. Such mathematical descriptions are notably hard to acquire, particularly for sub-sea systems. As explained by Al-Hajeri et al. (2009), reservoirs are typically kilometers underneath the seabed. With a limited range of available sensory equipment, modeling an oil basin based on first-principles requires interdisciplinary knowledge of, e.g., mechanics, seismography, stratigraphy, and petrophysics. Even if one obtains a highly accurate process model around the current operating point, it might not be accurate after a control change. The extraction process alters reservoir characteristics, such as pressures and composition of fluids. Induced transients and inevitably unmodeled disturbances create a need for online model calibration. This calibration is demanding due to a lack of available information, as we will revisit in the next chapter. If the plant-model mismatch is significant, well-known approaches like nonlinear model predictive control might fail (e.g., Jahanshahi et al. (2019)). This motivates *model-free* optimization methods.

Model-free optimization methods do not use an underlying plant model but instead use measurements to obtain gradient information about the process (Jäschke and Skogestad, 2011). Even though research on these approaches has received increased attention, human operators still perform most control changes in petroleum production systems. There are several reasons for this, some of which we include here:

- Production rates must be estimated following a control change. As measurements are affected by process dynamics, time delays, and uncertainties, the estimation typically happens over 1-4 hours, some time after a control change has been implemented.
- Operational changes are time-consuming. Implementation of a control change by a human operator might take hours, and transient dynamics can take several hours to stabilize. The production rate estimation may require several hours of measurements. Accordingly, we cannot expect to perform more than 1-2 optimization iterations per day.
- Risk of loss of revenue due to unmodeled behavior or sub-optimal production makes every control change of high potential value. Control changes should be kept small enough to mitigate this risk while maintaining a magnitude that causes a measurable effect.
- There might be a multitude of oil wells connected to the same topside processing facility, making well-specific production levels hard to estimate. It may not be desirable for a human operator to perturb all inputs at once to maintain an overview and control of the effects following a control change. However, not utilizing all control variables could slow down the optimization considerably.

Due to traditions in the petroleum industry that has been beneficially operating for decades, it is unlikely that any optimization method should be anything more than an assisting tool for the overseeing operator. The human-in-the-loop aspect implies that different company policies and individual operator habits lead to various control strategies being used in practice. Thus, there is likely room for increased efficiency. One example is utilizing all control variables due to the small number of optimization iterations per day. We illustrate in Figure 1.1 how simultaneous perturbations on a three-well system could increase production compared to sequential optimization. In general, many existing works on daily production optimization do not consider all of the above limitations. Therefore, it is desirable to search for optimization methods that make few underlying assumptions.

We will investigate the applicability of model-free, *stochastic optimization* methods to RTO of petroleum assets (Spall, 2012) . These are simple in nature, and the few underlying assumptions of these methods might make them suitable for local, iterative optimization. We will apply stochastic gradient search techniques for optimizing problems where only noisy meas-

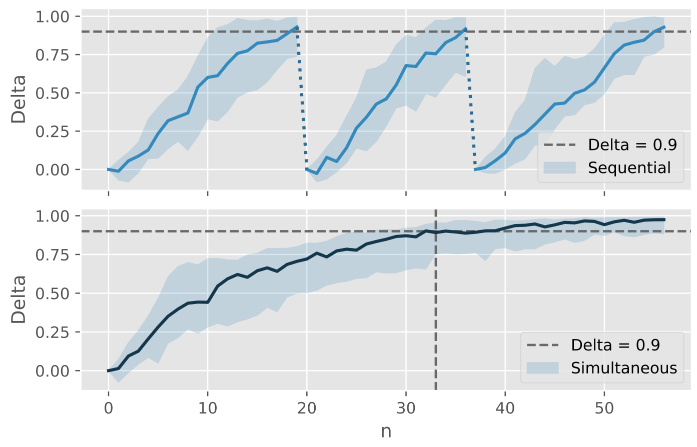


Figure 1.1: Sequential optimization of three different gas-lifted oil wells compared to optimizing on all wells simultaneously. Delta signifies production level and n is the iteration number. A simultaneous approach leads to 90% production in nearly half the amount of iterations.

urements of the objective are available. As we will later see, these methods form the basis for state-of-the-art *reinforcement learning* (RL) algorithms, where the maximization of a reward signal is at the core. As stated by Petsagkourakis et al. (2020, p. 1), these constitute "one of the few control techniques able to handle nonlinear stochastic optimal control problems." By optimizing directly on stochastic *decision policies*, these algorithms can solve problems that other machine learning methods fail to handle, such as mastering the game of Go. This is done by estimating a *policy gradient*; a direction of improvement for the decision policy that can increase the future rewards.

One major challenge with reinforcement learning is that successful applications often need vast amounts of data for learning (Sham M. Kakade, 2003; Schulman, 2016). For instance, tens of millions of games were used to train the AlphaGo algorithm. We will use the term *sample-efficiency* to denote the ability to find optimal control policies through a small number of optimization iterations, which more or less remains an open research topic. As previously established, each control change on a petroleum asset is time-consuming, accurate measurements are often unavailable, and modeling requires high maintenance and aptitude. Any petroleum optimization method should extract as much information possible from the noisy measurements. Furthermore, it should be able to deal with time-varying dynamics and physical constraints on the process. Finally, it should be feasible to implement online, respecting the previously listed limitations. With these requirements in mind, we propose the following research question:

> *Are policy gradient methods suitable for solving real-time production optimization problems?*

1.1 Scope and Research Objectives

Reinforcement learning techniques (as with most advanced control) can quickly get quite complex. A key concern in this thesis is to keep the methods as simple as possible. We will investigate the intersection between general stochastic gradient optimization and reinforcement learning. As the stochastic policy gradient estimates are exposed to high variance, we will experiment with a learned value function, essentially an estimate of the cost function. This is called an *actor-critic* approach. Although more complex methods are attractive and might yield better results in terms of production, we deliberately avoid these to examine the most basic underlying structures. This means we will not use adaptive optimizers or artificial neural networks to generate policies or value functions but rather strip the algorithms down to their basics. To confine the scope of the thesis, we define a set of research objectives that will help us answer the research question.

Primary Objective: Evaluate the potential of policy gradient methods on real-time production optimization.
Secondary Objectives:

- Present the theory needed to frame a production optimization problem as an RL task
- Construct a selection of optimization cases that reflect some of the challenges with daily production optimization
- Evaluate the sample-efficiency of some selected policy gradient algorithms on these cases
- Investigate the variance-reduction potential of the actor-critic framework

Using a simplified model for a system of gas-lifted petroleum wells, we aim to optimize a stochastic control policy in a minimal amount of iterations through stochastic gradient ascent. We will evaluate their performance by comparing results to the SPSA algorithm, which uses another stochastic gradient estimate based on finite differences. We review the applicability of the methods to real-world applications in terms of performance on these synthetic test cases, where we measure performance by the number of iterations needed for general convergence to optimal production.

1.2 Outline

This report is structured as follows. Chapter 2 summarizes the gas-lift problem and why this can be difficult. We briefly mention selected methods that have been used on this problem and further motivate our approach. In Chapter 3, we present the fundamental theory of stochastic gradient optimization and critical reinforcement learning concepts. These form the basis of the algorithms we present in Chapter 4 along with a description of the case studies. Here, we also describe the metrics used to evaluate the performances presented in Chapter 5. We discuss the results in Chapter 6, where we view them in light of real-time production optimization challenges. Finally, we conclude in Chapter 7.

Chapter 2

Background

In this chapter, we further motivate the research topic of this thesis, which is the use of policy gradient methods for petroleum optimization. As the literature on this is sparse, we first introduce more researched methods applied to the problem at hand; gas-lifted petroleum optimization. We then explain the fundamental idea behind reinforcement learning and current research on its use in the process industry.

2.1 Gas-Lifted Petroleum Optimization

One of the most commonly used forms of daily production optimization is a technique called *artificial gas-lift*. According to Ismail and Trjangganung (2014), about 70% of the global oil and gas production stems from maturing oil fields, with recovery rates averaging 35%, indicating possibilities of increased production. Over time, these maturing fields will deplete of oil, gas, and water, decreasing reservoir pressure. As the pressure might not be sufficient to lift the fluids to the surface, artificial lift methods have been used to counteract this for over a century (Jadid et al., 2006; Rashid et al., 2012).

One can support and maintain oil extraction by injecting pressurized natural gases into the petroleum wells, reducing fluid density and hydrostatic pressure drop in the well column. However, as the injection rate grows, the friction caused by additional gas flow might grow larger than the pressure reduction improvement, leading to decreased production. In this situation, an optimal injection flow rate for each well exists, maximizing the extraction rate and thus profits. Furthermore, the injection gas is often shared across wells (of which there are possibly several hundred) and of finite availability, making this a control allocation problem (Krishnamoorthy et al., 2019; Silva and Pavlov, 2020). Figure 2.1 shows a simple illustration of such a system. The functions relating the injection and production rates, referred to as gas-lift performance curves, are typically nonlinear and concave (Rashid et al., 2012). Therefore, optimizing the production would be equivalent to solving a nonlinear optimization problem on these curves, subjected to various constraints.

Optimal control would be easier if these curves were easily obtainable. In practice, that is not the case. As we have already established, physics-based modeling is intricate and requires precise data for model adjustments. However, the most reliable and easily obtainable production measurements are from after topside processing and thus after re-routing of wells to shared

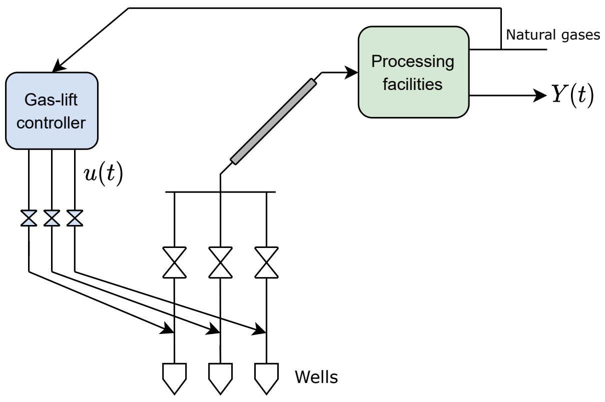


Figure 2.1: Simplified overview of a gas-lifted petroleum system

manifolds. Accurate oil rate measurements from individual wells would perhaps make model adjustments feasible, but these are not readily available. We will briefly elaborate on why, based on the handbook of Corneliussen et al. (2005).

A petroleum well produces water, gas, and slug in addition to oil. These phases are difficult to distinguish before they are processed in separators, which takes time. Accurate information about the phase flows is highly desired, making multiphase flow-metering a well-researched discipline within the petroleum industry. Two standard tools for obtaining this information are test-separators and multiphase flow meters.

The conventional way to characterize a specific oil well is through test separators, where phase volumes are measured based on fluid densities. Unfortunately, this requires re-routing a well flow, leading to reduced production, personnel intervention, and increased maintenance. Even if accurate rate measurements were obtained for single wells through the often limited amount of test-separators, finding optimal gas-lift allocation is nontrivial. As noted by Rashid et al. (2012), single-well methods fail to find optimal solutions when there is a network of wells. One reason for this is a pressure drop that arises downstream due to the routing of the wells into shared pipelines. To be able to monitor the entire network of wells accurately is essential for RTO. The newer approach to this is through the use of multiphase flow meters. Here, complex measurement instruments estimate the individual phase flows based on different principles. However, these costly flow meters are difficult to install, need to be calibrated regularly, and their accuracy is lower than for conventional methods (Corneliussen et al., 2005; Grimstad et al., 2021).

The impracticability of these techniques has impelled a search for new, *data-driven* methods that aim to utilize complex patterns in available data, as opposed to models based on first-principles. Several companies work towards data-driven optimization, including Solution Seeker. One approach to modeling is data-driven *virtual flow-metering*, where multiphase flow rates are estimated by modeling the underlying system without explicit prior knowledge (Grimstad et al., 2021). Virtual flow-metering is just one example of a wide array of ongoing research in applying machine learning principles in the industry (Mohammadpoor and Torabi, 2018; Sircar et al., 2021). Although showing promising results, research on data-driven modeling is still in development. Hence, there is a need for other model-free approaches to daily production optimization that do not heavily rely on individual measurements.

2.1.1 Selected Methods for Daily Production Optimization

A model-free optimization that has been widely researched for gas-lift optimization is *extremum seeking control* (ESC). By applying sinusoidal perturbations to the system, a gradient estimated from measuring the cost can be used to find an optimal input configuration (see, e.g., Dochain et al. (2011); Pavlov et al. (2017)). However, transient dynamics, long pipelines, and time delays make this difficult to implement. The standard ESC is a *local* optimization method, meaning it converges to local optima. Several of these might exist in a gas-lifted well network of many wells due to noise or local non-concave regions. With few daily optimization iterations and time-varying dynamics, global optimization without process models might be near impossible. Some research is done on making ESC more robust against sub-optimal solutions. Shu-Jun Liu (2012) introduces stochasticity in the perturbations to reduce chances of being stuck in local optima. By doing this, we move from *deterministic* to *stochastic* optimization.

Stochastic optimization is a general term that concerns optimization with noise present or where randomness is used in the search procedure itself (Spall, 2012). We focus on model-free methods that only use information acquired through noisy measurement of the optimization objective. We will delve into the details in the following chapter, but generally, a stochastic gradient can be used to optimize the *expectation* of a cost function. This approach is often referred to as *stochastic gradient* methods (Spall, 2012). These methods applied to the gas-lift problem were the main topic of research for the fall project preceding this thesis (Grepperud, 2021). They offer no guarantees of global solutions, but local solutions are likely the only possibility for real production assets.

One well-known stochastic gradient algorithm is *SPSA*, where a gradient is estimated by perturbing the process and measuring the output, based on the same principle as ESC (Spall, 1992). A few studies have discussed the use of SPSA in optimizing petroleum systems (Do et al., 2012; Hou et al., 2015; Wang et al., 2007). However, they either assume closed-loop optimization, i.e., an iterative two-step procedure, where one step updates prior geological knowledge, or consider different artificial lift methods. To the best of our knowledge, there exists no application of the SPSA algorithm on the gas-lifted optimal control problem. That is also the case of the *score function estimator* (Mohamed et al., 2019), another stochastic gradient estimate we will investigate. This estimator can be used to find the optimal distributional parameters of a probabilistic decision rule. Although widely used in machine learning, it is not commonly used in process industries.

Implementing stochastic gradients in practice is risky in applications such as petroleum optimization, where a fraction of reduction in production might lead to significant economic losses. Furthermore, oil wells can produce complex behavior during gas-lift. As stated by Dias et al. (2019), this could be in the form of highly oscillatory behavior, leading to sub-optimal production, unstable process behavior, and a loss of revenue. This suggests using advanced control strategies that stabilize the system and prevent intermittent production rather than stochastic gradient updates, which aim to optimize an objective immediately. In the process industry, *model predictive control* is the most popular technique for advanced control (Morari and H. Lee, 1999). However, the reliability is dependent on the accuracy of the model, which we have already established is complex to maintain.

Due to all of this, and perhaps due to a conservative industry, it is common for a human operator to oversee and perform daily production optimization. Sub-optimal control changes lead to loss of production, generating conservative practices, as explained in the introduction.

Senior operators use their best intuitions based on domain knowledge acquired through years of experience, transfer of knowledge, and learning by trial and error. As stated by Herbert Simon, "intuition is nothing more and nothing less than recognition" (Kahneman, 2011). With immediate access to large amounts of data, could an algorithm acquire such an intuition?

This motivates our investigation into reinforcement learning, which has received increasing attention over the last few years. The incredibly general structure of a RL problem consists of an *agent* performing actions in an *environment*, observing rewards and states along the way, illustrated in Figure 2.2. The historical development behind this class of machine learning is motivated by how humans and animals learn organically - by trial and error. The agent will seek to maximize its cumulative reward and learn what behavior is optimal over time. The combination of reward-based behavioral studies and optimal control theory has resulted in what is modern reinforcement learning (Sutton and Barto, 2020).

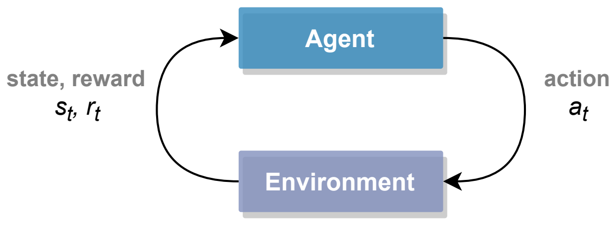


Figure 2.2: An agent interacts with an environment, observing rewards and states.

2.2 Reinforcement Learning for Industry Applications

With increasing computing power and significant advances in nonlinear function approximation through deep *artificial neural networks* (ANNs), reinforcement learning has shown great success in a wide variety of applications. Some examples include beating reigning world champions in the game Dota 2 (Berner et al., 2019), solving a Rubik's cube with a robotic hand (OpenAI et al., 2019) and mastering the game of Go (Silver, Huang et al., 2016). Although these examples use quite different approaches to achieve their goal, they all follow the same principles: learn a control policy that maximizes the reward function.

The theory on optimal control in RL mainly originate from early works on *dynamic programming* (DP) by Bellman (1957). Formulating the optimal control problem as discrete, stochastic models named *Markov decision processes*, dynamic programming was long acknowledged as more or less the only feasible way to solve such problems (Sutton and Barto, 2020). Many theoretical advances were on performing optimal decisions under stochasticity by assigning values to states and actions that reflect the reward function that is to be maximized. Suffering from what Bellman called "the curse of dimensionality," the significant advances in this approach to RL has only happened recently due to the abilities of deep learning to deal with extremely high dimensional spaces.

Much of the power of modern state-of-the-art RL methods lies in combining these *approximate* DP methods using general function approximators with numerical stochastic optimization. We focus more on the latter in this work, aiming to optimize a stochastic control policy through *policy gradient* methods. The flexible structure of the agent-environment interaction allows for incorporating model estimation into the stochastic optimization methods to accelerate learning. In this way, RL agents can learn complex, multi-stage control policies, striking a balance between planning and learning by trial and error (Schulman, 2016; Sutton and Barto, 2020). With the increasing attention on data-driven methods in the industry (Grimstad et al., 2021), RL methods could exploit the already growing infrastructures.

The practical applications of RL in the process industry are still yet limited, primarily due to the need to satisfy constraints (Pan et al., 2021). In most industrial systems, enforcing constraints is essential to ensure viable, economic, and safe performance. Consequently, the study of *safe* RL has become a discipline by itself; see e.g., Garcia and Fernandez (2015). Recent works show that it is possible to enforce constraints with high probability (Pan et al., 2021; Petsagkourakis et al., 2020), as well as guarantee safe policy updates (Chow et al., 2019).

Literature on applications of RL for daily production optimization is sparse. Ma et al. (2019) investigate deep RL methods on a water-flooded system but uses implicit policy derivations rather than direct policy optimization. Miftakhov et al. (2020) use policy gradient methods with pixel data, also on a water-flooded system; another method for artificial lift. To the best of our knowledge, there are yet no studies on gas-lift optimization using policy gradient methods. Some existing works, such as Andersen and Imsland (2021) and Gros and Zanon (2019), use the Q-learning algorithm to mitigate model uncertainty while using NMPC for the control policy. They utilize the learning of a model through approximate DP but do not employ policy gradient methods for optimizing the policy directly. We will introduce these methods formally in the following chapter.

Chapter 3

Theory

Reinforcement learning (RL) is an incredibly wide and continuously expanding research topic, making it very difficult to summarize in just a few pages. Although we cannot cover all the interesting details, we will go as far as to introduce actor-critic methods, which we will later use for production optimization. To justify the choice of methods, we will first make a brief review of model-free stochastic gradient optimization, primarily based on the theory from the preceding project assignment (Grepperud, 2021). We are interested in two ways of estimating a gradient that can be used in a stochastic gradient ascent-framework; A likelihood-ratio gradient based on the score function and a stochastic finite-difference gradient.

In the second part of the chapter, we shift our focus toward reinforcement learning and establish the underlying model; Markov decision processes. We explain key concepts and notation used to frame a process optimization problem as an RL task. We introduce the principles behind value-based learning and how they are used to estimate the underlying reward structures. We then explain direct policy optimization methods, which use stochastic gradients to improve the current control policy.

The final section focuses on how these two approaches are combined to form actor-critic methods. We will mention some variations of the "vanilla" policy gradient, as well as some ways to include constraints in the optimization problem. We will also explain the basis for natural policy gradients. Most state-of-the-art algorithms employ these, and we will also use these for our numerical studies.

3.1 Stochastic Gradient Optimization

A general probabilistic objective can be written as:

$$F(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{\theta})}\big[f(\boldsymbol{x})\big] = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} \tag{3.1}$$

where $\boldsymbol{\theta}$ are the distributional parameters of a probability density p and $\boldsymbol{x}$ is a random variable drawn from this distribution. We refer to a problem on the above form as a *mean-value problem* (MVP). The *sensitivity analysis* of F is

$$g = \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{\theta})}[f(\boldsymbol{x})], \tag{3.2}$$

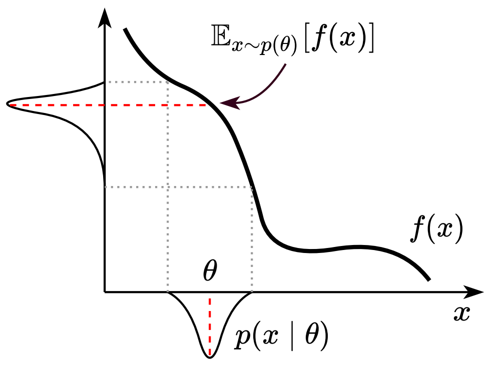


Figure 3.1: Optimizing expectations in a mean-value problem (MVP), with x sampled from a distribution $p(\theta)$

meaning it is the gradient of an expectation with respect to the distributional parameters. In this thesis, we are concerned with objectives on the form of eq. (3.1), where we aim to optimize the expected value of f because we do not have access to f itself. This is a central problem in many applications of machine learning (Buesing et al., 2016). This section and Section 3.1.2 are largely based on Mohamed et al. (2019).

A simple tool we can use to acquire knowledge about the objective is Monte Carlo estimators. By drawing independent samples $\boldsymbol{x}$ from the distribution $p(\boldsymbol{\theta})$, we can approximate the integral by averaging over the function evaluations at $\boldsymbol{x}$. As the number increases, we get an approximation converging to the true value by the law of large numbers. For our purposes, we can use Monte Carlo methods when estimating gradients that are unavailable on a closed-form, such as in eq. (3.2). These stochastic gradients can then be used to optimize an objective, such as the expectation of $f(\boldsymbol{x})$ in Figure 3.1.

3.1.1 Stochastic Gradient Descent

A simple model-based optimization method is gradient descent, where one iterates towards a minimum by moving in the direction opposite to the gradient. However, if the gradient is not available, one can use a stochastic approximation of the gradient to form *Stochastic gradient descent* (SGD):

$$\hat{\boldsymbol{\theta}}_{n+1} = \hat{\boldsymbol{\theta}}_n - \alpha_n \hat{g}_n, \tag{3.3}$$

Based on the stochastic approximation-framework of Robbins and Monro (1951), the iterative algorithm converges to local minima if α_n is chosen appropriately. We refer to Bottou et al. (2016) for more on convergence properties. If we can find an unbiased estimate of eq. (3.2), we can utilize this structure to optimize $F(\boldsymbol{\theta})$. An extension to SGD is to apply *momentum*, working as an exponentially moving average of the gradient estimate:

$$\boldsymbol{v}_{n+1} = \gamma \boldsymbol{v}_n - \alpha_n \hat{g}_n \tag{3.4a}$$
$$\hat{\boldsymbol{\theta}}_{n+1} = \hat{\boldsymbol{\theta}}_n + \boldsymbol{v}_{n+1} \tag{3.4b}$$

This acts as a smoothing factor, useful for gradient estimates with high variance. For the purposes of production maximization, we will consider stochastic gradient *ascent*, while still referring to the method as SGD. This is simply done by changing the subtraction in eq. (3.3) and eq. (3.4a) to form addition instead.

3.1.2 Score Function Gradient

For a probabilistic objective of the form eq. (3.1), where the function is only available through sampling, one can use a likelihood-ratio estimator to acquire the sensitivity analysis of eq. (3.2). This gives us the score function gradient estimator. The *score* of a probabilistic distribution is defined as the gradient of the log-likelihood function wrt. the distributional parameters (Wilks, 1962):

$$\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}} p(\boldsymbol{x} \mid \boldsymbol{\theta})}{p(\boldsymbol{x} \mid \boldsymbol{\theta})}. \tag{3.5}$$

Using this identity, the score function gradient can be derived:

$$\begin{aligned} g &= \nabla_{\boldsymbol{\theta}} \int p_{\boldsymbol{\theta}}(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} && \text{(3.6a)} \\ &= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{\theta})}\big[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x} \mid \boldsymbol{\theta}) f(\boldsymbol{x})\big], && \text{(3.6b)} \end{aligned}$$

with the full derivation and underlying assumptions explained in Appendix A. By sampling the system by drawing a stochastic variable from a distribution $\boldsymbol{x} \sim p(\boldsymbol{\theta})$ we get an unbiased Monte Carlo estimator for the expression above. We call it the *score function estimator*:

$$\hat{g}_P(\boldsymbol{\theta}) = \frac{1}{P} \sum_{p=1}^{P} f(\boldsymbol{x}_p) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_p \mid \boldsymbol{\theta}), \quad \boldsymbol{x}_p \sim p(\boldsymbol{x} \mid \boldsymbol{\theta}), \tag{3.7}$$

where P is the number of samples, which is often desired to be kept at a minimum for real-time optimization. In fact, we will later employ single-sample Monte Carlo estimates by letting $P = 1$, as they are still unbiased, although of higher variance.

To reduce the variance, likelihood-ratio gradients are often modified through the use of *baseline* models. Because the expectation of the score is zero, we can subtract an independent function b from $f(x)$ in eq. (3.6) while retaining unbiasedness:

$$g = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{\theta})}\big[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x} \mid \boldsymbol{\theta})(f(\boldsymbol{x}) - b)\big]. \tag{3.8}$$

3.1.3 SPSA

Another approach to estimating the gradient of an unknown function is through *finite-difference* (FD) methods. By evaluating a function *around* a point through multiple samples, a descent direction can be approximated. The *Simultaneous Perturbation Stochastic Approximation* (SPSA) algorithm uses a stochastic approach to form a FD gradient estimate. Introduced by Spall (1992), the algorithm consists of applying stochastic disturbances to the system and estimating a descent direction from measurements of the cost function. This direction can then be used in a Robbins-Monro framework such as eq. (3.3). Two opposite perturbations are used to estimate the gradient at iteration k, similarly to central FD methods:

$$\hat{g}(\hat{\boldsymbol{\theta}}_n) = \frac{y(\hat{\boldsymbol{\theta}}_n + b\boldsymbol{\Delta}_n) - y(\hat{\boldsymbol{\theta}}_n - b\boldsymbol{\Delta}_n)}{2b} \begin{bmatrix} \Delta_{n1}^{-1} \\ \vdots \\ \Delta_{np}^{-1} \end{bmatrix}. \tag{3.9}$$

The stochastic vector $\boldsymbol{\Delta}_n \in \mathbb{R}^M$ can be drawn from different distributions, but it should simultaneously perturb the current estimate $\hat{\boldsymbol{\theta}}_n \in \mathbb{R}^M$ in each of the M dimensions. Sadegh and

Spall (1998) suggest using a Bernoulli ± 1 distribution[1], i.e., every component of $\mathbf{\Delta}_n$ takes either a positive or negative value of 1 with equal probability. They argue that the Bernoulli distribution form is an optimal choice for an FD gradient approximation. Neither the normal or uniform distributions can be used with this FD approximation. This is because of infinite inverse moments, meaning their probability mass is centered around zero, so eq. (3.9) is undefined on average. The major advantage of this method is that only two samples are needed at each iteration, as opposed to $2M$ in a central FD estimate, but the same level of statistical accuracy can be achieved. A one-measurement SPSA estimate was proposed in Spall (1997), but is susceptible to higher variance. We focus on the two-measurement estimate, as earlier work found it outperforming the one-measurement method on a similar problem setup (Grepperud, 2021).

3.2 Reinforcement Learning Fundamentals

Reinforcement learning is a powerful, goal-oriented approach to computational intelligence, where an *agent* interacts with an *environment* to maximize a reward-signal. As the environment is generally unknown or stochastic, the agent must learn from feedback signals generated from the interactions with the environment, used to encourage or discourage certain future actions.

Although the current literature on RL often uses notation adopted from computer science (and from Sutton and Barto (2020) specifically), it is just one approach to the more general problem of optimal control. An agent, i.e., a *controller*, aims to find a control policy that optimizes some objective function. Model uncertainties or time-varying dynamics of the environment, i.e., the *process*, lead to what is essentially adaptive control. We will use notation and terms consistent with most RL literature, although most expressions have an equivalent in control theory literature (Lattimore and Szepesvari, 2020). We use capital letters for stochastic variables and lower case for realizations.

For purposes of petroleum production, we are focusing mainly on control policy optimization, where we are assuming that the underlying system is stochastic and not fully observable. As we saw in Chapter 2, modern reinforcement learning has origins in dynamic programming. Therefore, we formalize the general RL problem by introducing Markov decision processes, central to RL and the field *stochastic optimal control* (Lattimore and Szepesvari, 2020).

3.2.1 Markov Decision Processes

The standard reinforcement learning problem is expressed as a learning agent interacting with a *Markov decision process* (MDP) (Sutton, McAllester et al., 1999). MDPs are described by Sutton and Barto (2020, p. 47) as a "mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made." Introduced by Bellman (1957), an MDP is a tuple $\big(\mathcal{S}, \mathcal{A}, \Pr(s, s', a), R(s, s', a)\big)$ that denotes a discrete-time stochastic control process, with $\mathcal{S}$ and $\mathcal{A}$ being the state- and action-spaces, respectively. At each time step, an action $a \in \mathcal{A}$ will lead to a state $s' \in \mathcal{S}$ from the current state $s \in \mathcal{S}$ with a transition probability $\Pr(s, s', a)$. This transition yields a reward r from the reward distribution $R(s, s', a)$. The model satisfies the Markov property, i.e., memorylessness, so the next state s' is conditioned only on the previous state and action s, a (Howard, 1971). Because of this property, any optimal solution is a function of the current state. Unlike many applications of RL that consider MDPs

[1]also called a Rademacher distribution

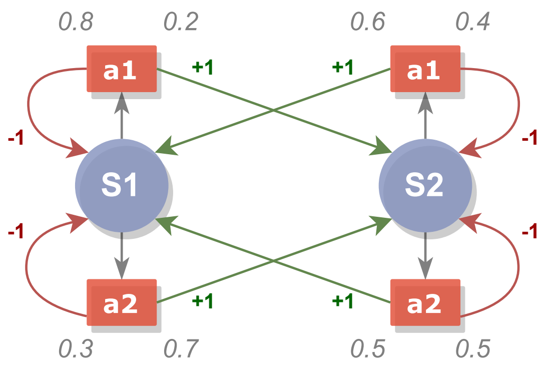


Figure 3.2: Simple stationary MDP with two possible actions in each of the two states. They lead to either a positive (green) or negative (red) reward with probabilities marked in grey.

with terminal states, we only focus on *non-episodic* settings, that is, continuing tasks where no such states exist. Although the model formally describes a fully-observable environment, almost all RL problems can be generalized as MDPs. This includes continuous state- and action spaces and partially observable problems (Silver, 2015), which leads to approximations of the solutions to these models.

A toy example of a simple MDP with two states *S1* and *S2* is shown in Figure 3.2. The actions *a1, a2* either lead to a new state, with a positive reward, or back to the current state, with a negative penalty. An RL agent aims to choose actions that over time maximizes the accumulated reward. In this stationary example, meaning the reward distribution does not change, the optimal actions are *a2* for state *S1*, and *a1* for state *S2*, as the expected *returns* are the highest for these actions.

Rewards, Returns and Policies

The objective of an agent is to maximize the cumulative reward attained from the reward-distribution $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ that might depend on states, actions, or both. The reward signal, called the reinforcement signal, is the driving force in the learning process. To maximize the reward, we thus let one maximization objective be the *expected return*, where the return G is some function of the cumulative rewards attained from each time step (Sutton and Barto, 2020). The returns can be designed according to the application, but two formulations are particularly common for continuing tasks: the discounted and average reward settings. At time step n, these two can be defined recursively:

$$G_n^\gamma = r_n + \gamma G_{n+1}^\gamma = \sum_{n=0}^{\infty} \gamma^n r_{n+k} \tag{3.10a}$$

$$\bar{G}_n = r_n - \bar{r}_\pi + \bar{G}_{n+1} = \sum_{k=0}^{\infty} r_{n+k} - \bar{r}_\pi \tag{3.10b}$$

For the discounted setting, $\gamma \in [0, 1)$ denotes the *discount factor*. Maximizing G_n^γ thus encourages immediate rewards over long-term future rewards, particularly useful in settings with a reward distribution that changes over time. With no terminal states, γ also ensures convergence of the series.

The average reward-formulation aims to maximize the difference between the next rewards and the average reward-rate, which we denote as $\bar{r}_\pi$, which in turn is equivalent to maximizing $\bar{r}_\pi$ itself. This is the expected rate of rewards attained from following the current control policy as time goes to infinity:

$$\bar{r}_\pi = \lim_{n\to\infty} \mathbb{E}_{a\sim\pi}[R_n \mid A_{0:n-1}] \tag{3.11}$$

An RL agent thus has a short-term objective in maximizing immediate rewards and a long-term objective, which is to maximize returns. We will now describe how this can be achieved – through policies and value functions.

A *policy* $\pi(a \mid s) : \mathcal{S} \to \mathcal{A}$ is a mapping from state to action, meaning it is a decision making rule for the agent. Thus, one way of solving an MDP would be to find an optimal policy that maximizes the reward. As done in Silver, Lever et al. (2014), we can write the performance objective as an expectation of the return from following a specific policy:

$$J(\pi) = \mathbb{E}_{a\sim\pi}[G(s,a)] \tag{3.12}$$

If the underlying system model was fully observable, such as in the toy example in Figure 3.2, this would simply reduce to computing the MDP analytically. In most situations, including this project, problem must be solved approximately. This leads us to two different main approaches to deriving RL algorithms, which are *policy optimization* and *value-based learning* (Schulman, 2016).

3.2.2 Value-Based Learning

Indirect RL, or value-based RL methods, aim to learn *value functions* through an approximate DP approach. Tracing its origins back to early work by Bellman (1957) and Howard (1960) amongst others, these methods assign values to states and actions. These are predictions of the reward the agent will receive, given a partially observable MDP (Schulman, Moritz et al., 2016). By estimating such values (which obey certain consistency conditions like the Bellman optimality equation, see Sutton and Barto (2020)), an optimal policy is implicitly derived by iterating over possible combinations of states and actions, and selecting actions with the highest value.

We are mainly interested in the state-value function $V(s)$, which models the value of a state s, i.e., the estimated future returns from following a policy π from s. The state-action-value $Q(a,s)$ is also important for decision making. It models expected return by choosing a specific action a given state s, and then following π:

$$V(s) = \mathbb{E}_{a\sim\pi}[G_n \mid S_n = s] \tag{3.13a}$$
$$Q(s,a) = \mathbb{E}_{a\sim\pi}[G_n \mid S_n = s, A_n = a] \tag{3.13b}$$

We have dropped the π-superscript in the above equations, commonly used to indicate a specific policy (Sutton and Barto, 2020). We only consider *on-policy* algorithms for the purposes of real-time production optimization, where the agent performs actions according to the best estimated policy. This is in contrast to off-policy algorithms such as Q-learning (Watkins, 1989) or DPG (Silver, Lever et al., 2014), where a target policy is typically updated from a different behavioural policy.

The functions of eq. (3.13) are unknown in a partially observable MDP and must be estimated. Historical approaches were often through linear estimators, but modern approaches typically utilize nonlinear approximators, such as ANNs. Q-learning is a prime example of an implicit algorithm showing solid results in discrete domains with deep RL (Duan et al., 2016; Mnih, Kavukcuoglu et al., 2013). For RTO, however, we will use value functions to assist the direct optimization of a stochastic policy. This leads us to another family of methods: direct reinforcement learning.

3.2.3 Policy Optimization

Policy optimization methods are the second major approach to RL, where a policy $\pi(\theta) = \pi_\theta$ is optimized directly with respect to the policy parameters θ, thus the name *direct* RL. As written by Schulman (2016), this frames the RL problem as a numerical optimization problem. Within policy optimization, we are focusing on *policy gradient* (PG) methods, which gradually update the policy parameters in an estimated direction of improvement.

Policy gradient optimization assumes stochastic policies, as the theory builds on probabilistic objectives as seen in Section 3.1. There is an inherent exploration as these policies outputs probabilities of choosing actions, unlike deterministic policies. This property is desirable, e.g., to avoid local optima, or because stochastic processes might lead to outcomes that differ over time. An advantage of these methods is that policy parametrization and approximation of a policy is often easier than value functions, particularly for large or continuous state spaces. PG methods also have stronger convergence properties than value-based methods by approximating gradient ascent and guaranteeing on-average policy improvement (Sham M. Kakade, 2003; Sutton, McAllester et al., 1999). Prominent historic approaches are FD methods and likelihood-ratio methods that use the score function estimator (Peters and Schaal, 2006). Most modern methods use the latter estimator. We will henceforth refer to these methods as "policy gradient methods", as they are derived from an important theorem.

Policy Gradient Theorem

The central theorem that enables likelihood-ratio PG methods is based on the score function gradient estimator (Section 3.1.2). The *policy gradient theorem* provides an analytical expression for how to change the parameters θ to improve the policy. Sutton and Barto (2020) presents the proof of the policy gradient, which for a continuous action space is:

$$\begin{aligned} g = \nabla_\theta J(\pi) &= \int_{\mathcal{S}} \mu(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) Q(s,a) da ds \\ &= \mathbb{E}_{a\sim\pi}\big[\nabla_\theta \log \pi_\theta(a \mid s) Q(s,a)\big], \end{aligned} \tag{3.14}$$

where $\mu(s)$ is the state distribution under π. This expression is nearly identical to eq. (3.6), where the Q-value is equivalent to the general function value $f(x)$.

One notable work within policy gradient methods is the REINFORCE algorithm of Williams (1992), where $Q(s,a)$ is estimated using episodic sample returns G_n^γ. This algorithm is known to suffer from very high variance and sample-inefficiency (Schulman, 2016) but has the property that the returns are bias-free in an episodic setting. Furthermore, it does not require careful generation of system-specific FD perturbation vectors such as SPSA (Peters and Schaal, 2006). To better separate good actions from great actions, one can reduce variance by introducing an arbitrary baseline function $b(s)$ in the gradient update, such as a state-value function $V(s)$. As long as it does not vary with a, the PG theorem remains valid (see Section 3.1.2).

REINFORCE utilizes sampled episodic returns[2], meaning the theory is based on finite MDPs (Sutton and Barto, 2020). A continuing implementation of this algorithm could be to view an iteration as an episode of length one so that the episodic returns equal reward at each time step. Initialization of a new episode, however, would introduce bias unless a new initial state is drawn at random. Letting the next initial state be the previous terminal state leads to what is essentially SGD with the score function gradient estimate (see Section 3.1.2). Another approach is to drop the notion of episodes altogether. We cannot approximate Q-values bias-freely, as we would need to look infinitely far into the future. This motivates *actor-critic* (AC) methods, that use uncertain predictions to estimate the current state- and state-action-values.

3.3 The Actor-Critic Framework

Actor-critic methods combine policy gradients with value-based learning, exploiting the advantages of each method to accelerate the learning. Two different models are used: the *actor*, which is the policy model performing actions, and the *critic*, which typically is a value function. The critic evaluates the action done by the policy by comparing the observation (r, s') to the predictions (Q, V). This yields a scalar value used to update subsequent estimates and inform the actor how to modify the policy. This form of bootstrapping is called *temporal-difference* (TD) learning, and is the difference between AC-methods and PGs with value-function baselines. The *one-step* TD-error is defined by Sutton and Barto (2020) as

$$\delta^{\gamma} = r + \gamma V(s') - V(s) \tag{3.15a}$$

$$\bar{\delta} = r - \bar{r}_{\pi} + V(s') - V(s) \tag{3.15b}$$

for the discounted- and average-return settings, respectively. The one-step TD-error can be seen as special case of an *advantage function* estimate: $A(s,a) = Q(s,a) - V(s)$ (Mnih, Badia et al., 2016). The Q-value is approximated with $r_n + \gamma V(s')$ for the discounted case, which is equal in expectation:

$$\begin{aligned} Q(s,a) &= \mathbb{E}_{a\sim\pi}[G_n \mid S_n = s, A_n = a] \\ &= \mathbb{E}_{a\sim\pi}[r_n + \gamma G_{n+1} \mid S_{n+1} = s'] \\ &= \mathbb{E}_{a\sim\pi}[r_n + \gamma V(s') \mid S_{n+1} = s']. \end{aligned}$$

The advantage function is a measure of how good an action actually is, compared to an earlier prediction, and is a generalization of the prediction-correction structure in eq. (3.15). If an optimal policy is known, every optimal action will yield a Q-value $Q(s,a)$ that is equal to the state-value $V(s)$ at that iteration; the maximum value of $A(s,a)$ is zero. During learning, however, A indicates if an action was better than expected, and this is used to update the policy parameters. As shown by Sutton, McAllester et al. (1999), an advantage estimate like the TD-error can be used directly in eq. (3.14) to optimize a policy.

3.3.1 Variations of the Policy Gradient

The policy gradient estimate can be used in a stochastic gradient ascent framework (eq. (3.3)) with

$$\hat{g} = \nabla_{\theta} \log \pi_{\theta}(a \mid s)\psi(\cdot) \tag{3.16}$$

[2]Referred to as "Monte Carlo" returns by Sutton and Barto (2020), specifically meaning averaging *complete episodic returns*, not to be confused with a general Monte Carlo estimator like in Section 3.1.2

as a Monte Carlo estimator and ψ indicating some estimate of $Q(s,a)$. As we have seen, ψ might be a sampled episodic return, a one-step TD-error, or lower-variance advantage estimates. Algorithms utilizing advantage estimates have been increasingly popular, as it "yields almost the lowest possible variance" for policy gradient updates, as stated by Schulman, Moritz et al. (2016, p. 3). In their paper, they propose the GAE method, which is a common advantage estimator choice for state-of-the-art AC algorithms. Here, they sample trajectories of observations before updating the estimate (so-called batch methods). A well-known example of batch updates and advantage estimates for policy optimization is the *Advantage Actor-Critic* (A2C) algorithm (Mnih, Badia et al., 2016), using *n-step* TD-updates from sampled trajectories. Batch methods reduce variance in the policy gradient estimation, approximating the P-sample score function estimator we saw in eq. (3.7).

The variations listed above are often referred to as "vanilla" PG (OpenAI, 2018), because they do not modify the score function. Table 3.1 shows a selection, including the single-sample score function estimate Score1, which we return to in Section 4.2.1.

Table 3.1: Variations of the standard Policy Gradient

Variation	Q-estimate
Policy Gradient Theorem	$\psi_n = Q(s,a)$
REINFORCE	$\psi_n = G_n$
REINFORCE w/ baseline	$\psi_n = G_n - b_n$
Score1 w/ baseline	$\psi_n = y(a_n) - \bar{r}$
Advantage Actor-Critic	$\psi_n = Q(s,a) - V(s)$
Discounted TD-AC	$\psi_n = r_n + \gamma V(s') - V(s)$
Average-Reward TD-AC	$\psi_n = r_n - \bar{r}_\pi + V(s') - V(s)$

3.3.2 Natural Policy Gradients

One property of the score function gradient estimator is that its variance is the Fisher information (Mohamed et al., 2019). This has sub-optimal implications for Gaussian policies, a common choice for continuous RL problems. As stated by Chou et al. (2017, p. 5), the variance of a Gaussian PG estimator is inversely proportional to the distributional variance σ^2: "as the policy improves and becomes more deterministic, the variance of [eq. (3.7)] goes to infinity." We may address this with *natural policy gradients*, which adjust the learning rate according to the distribution. Since we are optimizing on a distributional parameter space with a different structure than the gradient of the objective itself, the traditional gradient ascent does not always yield the steepest direction. Natural gradients, however, do precisely this by using the Fisher information $\mathcal{I}(\theta)$ rather than Euclidean distance as a metric (Amari, 1998; Sham M Kakade, 2001):

$$g^{nat} = \mathcal{I}^{-1}(\theta) g \tag{3.17}$$

where the Fisher information is

$$\mathcal{I}(\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s) \nabla_\theta \log \pi_\theta(a \mid s)^\top\right]. \tag{3.18}$$

This approach is analogous to the second-order Newton's method in that the Fisher information can be derived as the Hessian of the Kullback-Leibler (KL) divergence between two distributions, as stated by Martens (2014). The natural policy gradient g^{nat} grants a more accurate local approximation than the standard PG g of eq. (3.14), as it contains information about the curvature of the parameter space. Although the Gaussian PG yields the correct direction, g^{nat} uses this information to find better step lengths (Chou et al., 2017). Unlike Newton's method, natural gradients do not assume a locally-quadratic cost-function (Amari, 1998) and they are invariant to the policy parametrization (Martens, 2014).

State-of-the-art AC algorithms tend to utilize natural policy gradients. The TRPO (Schulman, Levine et al., 2015) and PPO (Schulman, Wolski et al., 2017) algorithms are based on constraining the KL divergence between policy updates to restrict arbitrary large updates. These trust region methods are proved efficient for continuous control, especially on locomotion tasks (Duan et al., 2016). A more sample-efficient trust region method is the ACKTR algorithm that optimizes value functions using Gauss-Newton approximation (Wu et al., 2017).

3.3.3 Constrained Policy Optimization

In most real-world control applications, there are often physical limitations to a system, enforcing constraints on the optimization problem. An example could be the maximum torque applied to a robotic arm or the total available injection gas for a gas-lifted petroleum well. Inherent stochasticity in the environment might make even optimal policies perform poorly in some aspects, such as concerning risk and safety (Heger, 1994). There exists much literature on constrained stochastic optimization and RL (see, e.g., Garcia and Fernandez (2015)), so we only consider some intuitive methods that are easy to implement: augmented objectives, clipped policy gradients, and bounded policies.

For applications where constraints are not safety-critical and must only be satisfied most of the time, augmenting the reward-function might be a feasible approach. Although *reward-shaping* is generally used to counter sparse reward signals (Hu et al., 2020), it can also be used to guide the agent by penalizing constraint violations. This method works in a similar way to barrier-methods in numerical optimization that transform a constrained optimization problem to an unconstrained one (Nocedal and Wright, 2006). Policy gradient methods using augmented objectives are often referred to as Lagrangian methods (Chow et al., 2019). Adjusting the rewards is a common technique, and is often combined with only allowing small policy updates through trust-region methods (Petsagkourakis et al., 2020).

A Gaussian policy is unbounded in its support, meaning the policy output has a non-zero probability of being an action outside allowed bounds. In many popular algorithms, the environment clips the action within these bounds, but the agent estimates the policy gradients as if the actions were not clipped, introducing bias (Chou et al., 2017; Duan et al., 2016). Addressing this, Fujita and Maeda (2018) proposes the unbiased, lower-variance Clipped Action Policy Gradient (CAPG) estimator, where the score function-term $\zeta(s,a,\theta) = \nabla_\theta \log \pi_\theta(a \mid s)$ in eq. (3.16) is replaced with

$$\bar{\zeta}(s,a,\theta) = \begin{cases} \nabla_\theta \log \Pi_\theta(a_{\min} \mid s) & \text{if } a \leq a_{\min} \\ \nabla_\theta \log \pi_\theta(a \mid s) & \text{if } a_{\min} < a < a_{\max} \\ \nabla_\theta \log(1 - \Pi_\theta(a_{\max} \mid s)) & \text{if } a_{\max} \leq a \end{cases} \tag{3.19}$$

This can be used in combination with any of the variations seen in Table 3.1, and the bounds can also be specified to a specific region around the current state, inducing a trust-region scheme. The intuition behind CAPG is that, for a policy where actions are clipped within bounds by the environment, the agent will observe similar rewards and thus have no information on which direction to move the policy. Figure 3.3 illustrate how the cumulative density Π_θ is used when an action is too large.

An alternative approach to addressing bias introduced by ignoring action bounds is to use bounded policies. Chou et al. (2017) suggests using the Beta distribution rather than the standard Gaussian for continuous control, as it takes values between zero and one. This method has shown promising results combined with PPO (Petrazzini and Antonelo, 2021) and is quite simple in nature. However, Gaussian policies remain the standard choice for continuous control.

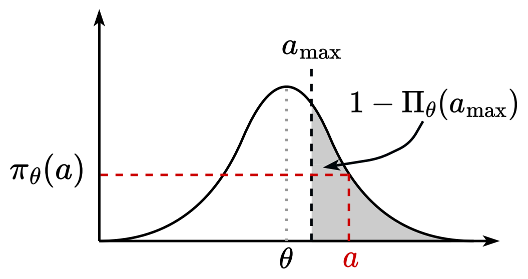


Figure 3.3: The CAPG estimator uses the cumulative density if an action is outside bounds.

Chapter 4

Method

This chapter describes the process model used for numerical simulations. We describe assumptions put on the model and how this will affect the algorithms used. We then explain the three main optimization algorithms based on stochastic gradient search, where the system is subjected to perturbations to form gradient estimates. We compare a one-step policy gradient agent to a one-step average-reward AC agent using SGD and the SPSA algorithm as a benchmark.

4.1 Case Study - Gas Lifted Petroleum Asset

Our study uses a simplified model of an artificially gas-lifted sub-sea petroleum system based on the model of Silva and Pavlov (2020). As described in Chapter 2, gas-lift refers to the action of injecting gases into petroleum reservoirs to help lift fluids from the seabed. The injected flow is controlled by a valve which is called the gas-lift *choke* and is transported down to the reservoir through the *annulus*, which reduces hydrostatic pressure in the *tubing* of the well. Such a system is illustrated in Figure 4.1, courtesy of Imsland (2002). The input-output relation between injected gas and produced oil can be expressed through *gas-lift performance curves*, also called production curves (Silva and Pavlov, 2020).

Our simulation study considers a system of $M = 5$ petroleum wells, each with an individual gas-lift choke. The plant input is a vector of the form $\boldsymbol{u} = (u_1, \dots, u_5)$, where each component is the gas flow $\mathrm{kg\,s^{-1}}$ injected into the annulus. The production curves are modeled as concave functions of the form

$$f_i(u_i) = c_{1,i} \cdot 10^{-7} \cdot u_i^4 + c_{2,i} \cdot 10^{-4} \cdot u_i^3 + c_{3,i} \cdot 10^{-2} \cdot u_i^2 + c_{4,i} \cdot u_i + c_{5,i}, \tag{4.1}$$

with $c_{j,i}$ being production curve coefficient j for well i, unknown to the optimization agent. The output of the plant is $F = \sum f_i$. We have modified the coefficients from Silva and Pavlov (2020) slightly to create performance curves with different characteristics, as well as letting them change over time specifically for a tracking case. Due to topside processing of the oil flow, we let the only available measurement be the total oil production flow:

$$Y = \sum_{i=1}^{M} f_i(u_i) + \eta, \quad \eta \sim \mathcal{N}(0, \sigma_{\text{noise}}) \tag{4.2}$$

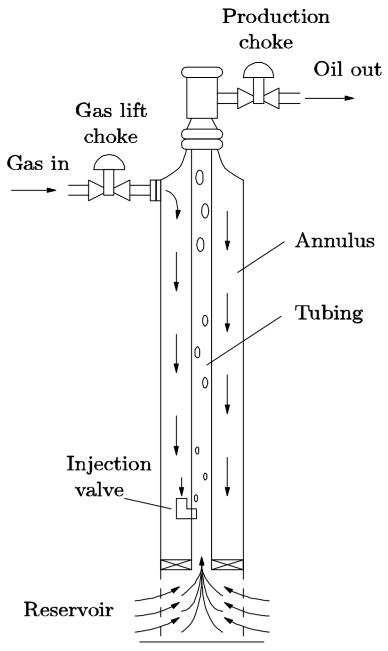


Figure 4.1: Illustration of a gas-lifted sub-sea petroleum well (Imsland, 2002).

The expectation of Y is thus F. Based on this, we define a probabilistic objective function that is to be maximized:

$$J(\boldsymbol{u}) = \mathbb{E}_{\boldsymbol{u}\sim\pi_\theta}\left[\sum_{i=1}^{M} f_i(u_i)\right] \tag{4.3}$$

4.1.1 Optimization Considerations

As motivated in Chapter 1, several challenges with real-time production optimization should be considered, such as the process complexity and the currently existing control practices. As previously mentioned, the only output measurement available is of the total production flow, Y. As described in Chapter 2, rate estimation for individual wells is an open research problem, so we will assume that only the total oil flow is measurable. For simplicity, we add white noise to the measurements, although these are subjected to various disturbances and in practice. The single output measurement Y means that, for a system with M wells and internal states f_i, it is (locally) observable only when $M = 1$ (Krener and Ide, 2009). We should thus assume that variance in estimation and optimization performance will increase with dimensionality.

Every optimization iteration for a petroleum asset is costly in terms of potential loss of revenue and time. Due to transient dynamics, transport, and process time, as well as delayed implementation due to the human-in-the-loop assumption, we assume that only two optimization iterations can be done per day. Input perturbations should generally be kept small enough to be safe but large enough to cause a measurable effect. Domain knowledge should thus be used when designing the search variance for stochastic control policies. Observability of the system also affects how it is controlled, with operators presumably perturbing fewer control variables simultaneously. However, we will utilize all inputs to increase efficiency, as illustrated in Figure 1.1.

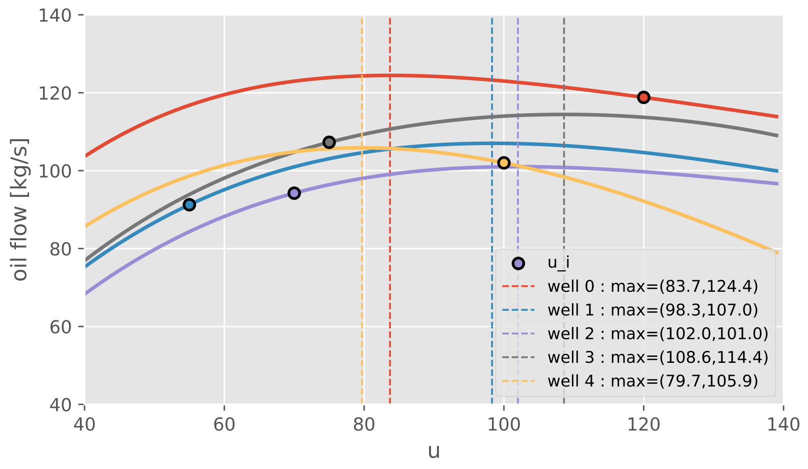


Figure 4.2: Gas-lift performance curves with $M = 5$ wells. Circles mark the initial input configuration. Dashed lines indicate optimal input-output pairs for each well.

The production curves are only valid in the interval of $40 \leq u_i \leq 140\,\mathrm{kg\,s^{-1}}$. In all simulations, these individual input constraints are enforced on the controller by clipping policy outputs into allowed bounds. We address this in Section 4.2. There might exist a multitude of other constraints on a real petroleum system, such as the handling capacity of water, sand, and various gases that are by-products of oil production. Water-handling capacity constraints is considered in Silva and Pavlov (2020). Furthermore, gas-lifted production might also be modeled as an economic optimization problem (Jahanshahi et al., 2019) due to gas-lift cost. Instead, we are focusing on injection gas as a limited resource in Case 3 (Section 4.1.4), making this a control allocation problem.

Finally, the underlying dynamics are hard to model, which is the primary motivator for exploring stochastic optimization. These dynamics include depletion of the reservoir, where more gas must be injected to maintain production. Optimization algorithms should therefore have the ability to track a time-varying optimum. As described in Chapter 2, the output is not necessarily a linear sum of the wells. Another phenomenon affecting production is the covariant dynamics that might arise due to pressure transients in interconnected pipelines. However, we do not consider this in our test cases.

We summarize the assumptions and simplifications made:

(i) Available measurements are the inputs u_i and total production Y
(ii) Control changes are not restricted in dimension
(iii) Control changes should be kept large enough to produce an observable change
(iv) Control changes should be kept reasonably small to avoid risk
(v) Inputs must be within allowed bounds; $40 \leq u_i \leq 140$
(vi) There exists a maximum available injection amount $U^{\max}$ (Case 3 only)
(vii) The optimization horizon is $N = 120$, corresponding to two iterations per day for 60 days

Table 4.1: Static performance curve coefficients

	$c_{1,i}$	$c_{2,i}$	$c_{3,i}$	$c_{4,i}$	$c_{5,i}$
well 1	-3.9	2.1	-4.3	3.7	12.0
well 2	-1.3	1.0	-2.8	3.1	-10.0
well 3	-1.2	1.0	-2.8	3.1	-17.0
well 4	-4.0	1.8	-3.6	3.5	-16.0
well 5	-1.4	1.0	-2.9	3.0	6.0

Table 4.2: Dynamic performance curve coefficients

	$c_{1,i}$	$c_{2,i}$	$c_{3,i}$	$c_{4,i,\text{start}}$	$c_{4,i,\text{end}}$	$c_{5,i,\text{start}}$	$c_{5,i,\text{end}}$	n_{jump}
well 1	-1.4	1.0	-2.9	2.9	3.7	16.0	-80.0	0
well 2	-3.9	2.1	-4.3	3.2	3.8	42.0	-18.0	10
well 3	-1.3	1.0	-2.8	2.6	3.2	20.0	-50.0	30

4.1.2 Case 1: Static Optimization

The simplest optimization case study consists of static performance curves in the form of eq. (4.1). The coefficient values are found in Table 4.1. The corresponding performance curves and a five-dimensional input configuration are seen in Figure 4.2. We let there be no system-wide constraints on injection gas. As we only have the measurements of the total oil flow Y and input $\boldsymbol{u}$, we will first investigate how observability affects the controller's performance. We compare simulations with $M = 1, 3, 5$ wells, respectively. As dimensionality increases, the single-sample gradient- and value function estimates will likely suffer from more variance, as each sample contains relatively less information.

We define three noise-settings for the simulations, affecting eq. (4.2). The *no-noise* setting with $\sigma_{\text{noise}} = 0$ yields a measurement $Y = \sum f_i$. In the *low-noise* setting, production measurements are affected by Gaussian noise with a standard deviation (SD) σ_{noise} of 0.1% of the production value F. In the *high-noise* setting, we let $\sigma_{\text{noise}} = 1\%$ of F. The low-noise setting is what we mainly use to compare performance across algorithms. We use no- and high-noise settings to compare robustness under noise.

4.1.3 Case 2: Tracking

To simulate a reservoir depleting over time, we have modified the well-curve coefficients to reflect the increasing amount of injected gas required to keep production at an optimum. The coefficients $c_{4,i}, c_{5,i}$ start from an initial value and follow a linear trajectory over time. The optimum moves non-linearly, simulating a tracking problem. Note that the dynamics are exaggerated, with optimal injection rate nearly doubling over time. Additionally, we have implemented well-specific time delays before the curves change. These delays aim to simulate that a petroleum system might experience unpredictable and abrupt disturbances of varying scales. In our model, this corresponds to a jump at different time steps. This is visualized in Figure 4.3. We are only investigating a system with one and three wells without noise for this case to isolate the effects of a moving optimum. Coefficient values are shown in Table 4.2, along with the iteration number where the optimum starts to move.

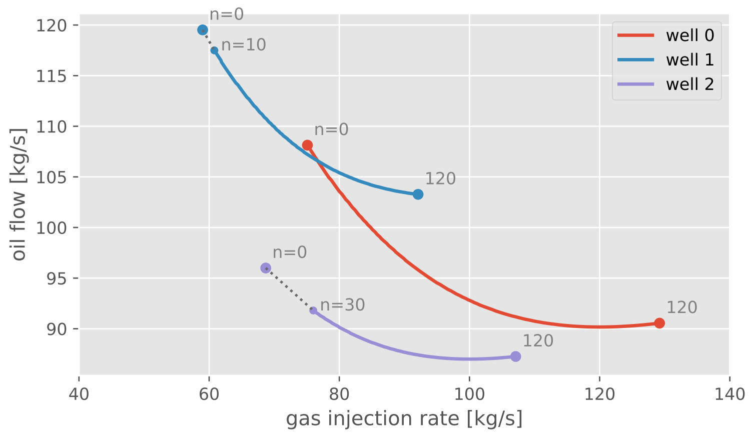


Figure 4.3: Optima of well curves that change over time with $M = 3$ wells. The optimum jumps at $n = 10, 30$.

4.1.4 Case 3: Tracking with System-Wide Constraints

For the final case study, we extend the tracking model where an increasing amount of gas is needed to maintain optimal production. We choose to model a system-wide constraint on available input, $\sum u_i < U^{\max}$, so that the constraint becomes active at a point during the simulation. Our approach is to augment the objective function so that maximization of the augmented objective will enforce constraint activation. In reinforcement learning terms, this is called reward-shaping. Since our algorithms will use SGD, we cannot use a log-barrier method such as in Silva and Pavlov (2020), as it may lead to infinitely large gradient updates. Instead, the production level seen by the agents has a linear decay that is proportional to the injected gas that surpasses $U^{\max}$. The new objective becomes:

$$J^{\text{aug}}(\boldsymbol{u}) = \begin{cases} \mathbb{E}_{\boldsymbol{u}\sim\pi_\theta}\left[U^{\max} + \sum_{i=1}^{M} f_i - u_i\right] & \text{if} \quad \sum u_i \geq U^{\max} \\ J(\boldsymbol{u}) & \text{otherwise} \end{cases} \tag{4.4}$$

A visualization is shown in Figure 4.4 for a single-well system. As before, we use $M = 1, 3$. We use $U^{\max} = 90$ for the single-well system, and $U^{\max} = 80 \cdot M$ for the three-well system.

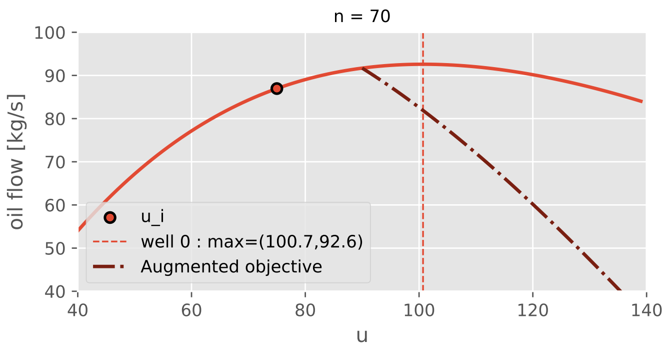


Figure 4.4: Single-well system at $n = 70$ with augmented objective. $U^{\max} = 90$. The constraint becomes active at the optimum at $n = 45$.

4.2 Algorithms

In order to frame the production optimization problem as an RL problem, we first define some key components. As we are maximizing oil flow, we can let the reward function be the measurement of eq. (4.3). The reward function is intentionally kept simple to avoid "reward-hacking" (Amodei et al., 2016) and to keep it similar to a traditional stochastic optimization objective. For the constrained case in Section 4.1.4, we let the reward function be the measurement of the augmented objective function of eq. (4.4). One advantage of this choice is that the agent will observe rewards at every time step and thus hopefully learn how to improve. Following notation from Chapter 3, the action a is the input to the system $\boldsymbol{u}$. The state s may be defined in several ways. We let s simply be the current input configuration, $s = \boldsymbol{u}$.

The nature of the objective function and the symmetric policy distributions makes it so that maximizing production roughly translates to finding an optimal mean value. All algorithms are implemented with the momentum SGD framework of Section 3.1.1 with a weight factor of γ to equalize variance in the gradient updates. Flowcharts of the algorithms are presented in Figure 4.6.

4.2.1 Score Function Gradient

The first algorithm is based directly on the score function gradient estimator, using single rewards as an approximation to the Q-value in eq. (3.14). To reduce variance, we also use a baseline in the form of an average-reward rate $\bar{r}$, gradually updated at each iteration with a weight of η. As the action space is continuous, we use a standard diagonal Gaussian, behaving similar to M univariate Gaussians, meaning that we estimate an optimal SD σ_i and mean μ_i for every input dimension. This assumes that each well is decoupled, which is a simplification only in the case of system-wide constraints. To ensure that $\sigma > 0$, we map the SD to a variable s by using a *softplus* function with a small offset $c > 0$ and a weight factor of β:

$$\sigma_i = \frac{1}{\beta} \log(1 + \exp(\beta \cdot s_i)) + c$$

This leads to a different analytical expression for the Gaussian score function gradient, but it behaves similarly, as shown in Appendix A.2. We initially experimented with both softmax- and exponential mappings but found that the linearity of the softplus makes it easy to work with. The policy parameters are then $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{s})$.

We have used two different methods to compute the policy gradient. One method will be using the CAPG estimator from Section 3.3.3. Here, the agent is aware of the individual input bounds imposed by the environment and uses the cumulative density function when an action is sampled out of bounds. During simulations, we found this was effective when policy updates were small, and the mean was close to the bounds.

The other method is to use natural policy gradients (see Section 3.3.2). This means that the Fisher information must be estimated, which may be done by directly sampling the policy gradient, as can be seen from eq. (3.18). However, for a Gaussian policy, the expected value can be analytically derived. Chou et al. (2017) shows that for a Normal distribution, where $l(\theta) = \log \pi_\theta(a \mid s)$, this is

$$\mathcal{I}(\theta) = -\mathbb{E}_{a\sim\pi}\left[\frac{\partial^2 l(\theta)}{\partial \theta^2}\right] = \begin{bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{2}{\sigma^2} \end{bmatrix}. \tag{4.5}$$

In practice, this is will simply scale the policy gradient with factors of σ_i^2. The gradient of the Gaussian score estimate is

$$\nabla_{\theta_i} \log \pi_{\theta_i}(a_i) = \left[\frac{a_i - \mu_i}{\sigma_i^2} \quad \frac{1}{\sigma_i}\left(\frac{(x_i - \mu_i)^2}{\sigma_i^2} - 1 \right) \right], \tag{4.6}$$

so the inverse Fisher partially cancels out variance terms in the denominators. The respective derivations of Equations (4.5) and (4.6) are shown in Appendices A.3 and A.2.

Algorithm 1 shows the workflow of the algorithm using natural gradients. We name this Score1 as it uses a single-sample Monte Carlo score function estimator. CAPG may replace the natural gradients in Algorithm 1 by substituting the inverse Fisher and score function with $\bar{\zeta}$ from eq. (3.19).

Algorithm 1 Score1 with Natural Gradient and Average-Reward Baseline

```
Require: Differentiable policy $\pi_\theta$, step sizes $\alpha_\theta, \eta > 0$
  for $n = 1, 2, \ldots$ do
      Agent samples action $a$ from policy, observing reward $r$ and next state $s'$:
          $a \sim \pi_\theta(s)$
      Compare reward with baseline:
          $\delta = r - \bar{r}$
      Compute Fisher information $\mathcal{I}(\theta)$
      Update policy parameters using $\delta$:
          $\theta \leftarrow \theta + \alpha_\theta \mathcal{I}^{-1} \nabla_\theta \log \pi_\theta(a \mid s)\delta$
      Update average reward using $\delta$:
          $\bar{r} \leftarrow \bar{r} + \eta\delta$
  end for
```

4.2.2 Average-Reward Actor-Critic

Following an actor-critic approach, we extend Score1 by using a value function estimate to model the reward distribution. Since our objective is production maximization, we are interested in immediate rewards. The path to the optimum itself is less important than arriving there quickly. An option would be to employ a heavily discounted one-step TD-error for the policy updates. However, Sutton and Barto (2020) show that the discounted return setting can be deprecated for continuing tasks and suggest using the average-reward setting instead. Recall the one-step TD-error for the average-reward formulation:

$$\bar{\delta} = r - \bar{r}_\pi + V(s') - V(s).$$

As we have defined s as the current control figuration u, the first two terms can be seen as a measured error and the last two as predicted error. The terms $\bar{r}_\pi + V(s)$ act as a baseline, independent of the action taken. As we do not know $\bar{r}_\pi$, we learn it gradually from $\bar{\delta}$ with a step size of η. This algorithm uses natural policy gradients as they showed promising results during implementation of Score1. The Average-Reward Actor-Critic algorithm is presented as Algorithm 2. In practice, this algorithm is very similar to Algorithm 1. They differ only in adding a value function, which should accelerate the learning, given that it is modeled adequately.

Algorithm 2 Average-Reward Actor-Critic

```
Require: Differentiable policy $\pi(\theta)$, value function estimate $V_w(s)$, step sizes $\alpha_\theta, \eta > 0$
  Initialize $V$ by perturbing the system around the initial state
  for $n = 1, 2, \ldots$ do
      Actor samples action $a$ from policy, observing reward $r$ and next state $s'$:
          $a \sim \pi_\theta(s)$
      Critic calculates TD-error based on value function:
          $\bar{\delta} = r - \bar{r} + V(s') - V(s)$
      Compute Fisher information $\mathcal{I}(\theta)$
      Update actor's policy parameters using $\bar{\delta}$:
          $\theta \leftarrow \theta + \alpha_\theta \mathcal{I}^{-1} \nabla_\theta \log \pi_\theta(a \mid s)\bar{\delta}$
      Update critic weights by fitting the last $p$ samples:
          $w \leftarrow \text{LLS}\{r, a\}^{(p)}$
      Update average reward using $\bar{\delta}$:
          $\bar{r} \leftarrow \bar{r} + \eta\bar{\delta}$
  end for
```

Value Function Estimation

The value function models the reward distribution of the process as a function of states and actions. With the reward defined as measured production, this roughly translates to modeling the performance curves. Although estimation of these curves is a research topic by itself, domain knowledge suggests that they are generally concave, smooth, and well-behaved (Rashid et al., 2012). By putting these basic assumptions on the underlying model, we have chosen to use a quadratic polynomial as a function approximator rather than using an artificial neural network (ANN). Their high capacity typically requires larger amounts of data to be accurate. A quadratic polynomial was hypothesized to give a local approximation good enough to accelerate policy updates.

The value function is estimated by using model history, i.e., actions and rewards, to fit the data by using linear least squares (LLS) methods with L2-regularization. For a system of M wells, a quadratic polynomial will be of $d = 2M + 1$ dimensions. As stated by Conn et al. (2009), a minimum of $p = d + 1$ samples is needed for a unique solution to exist. Such a solution might not exist, however, if the system cannot fully be described by the linear polynomial, which is the case here. Our approach is then to sample more data, and to fit the p samples with the ordinary least squares solution to the system

$$\begin{bmatrix} 1 & a_1^0 & \ldots & a_d^0 \\ 1 & a_1^1 & \ldots & a_d^1 \\ \vdots & \vdots & & \vdots \\ 1 & a_1^p & \ldots & a_d^p \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} y^0 \\ y^1 \\ \vdots \\ y^p \end{bmatrix}, \tag{4.7}$$

which is an optimal approach with normally distributed sample noise, by the Gauss-Markov theorem (Gentle et al., 2012). For M wells, this yields a polynomial of the form $\hat{y}(a) = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_d a_M^2$. L2-regularization is applied on $w_i, i \neq 0$ with a penalty factor of λ, in order to keep the curvature low. This is particularly useful when the sample variance is low,

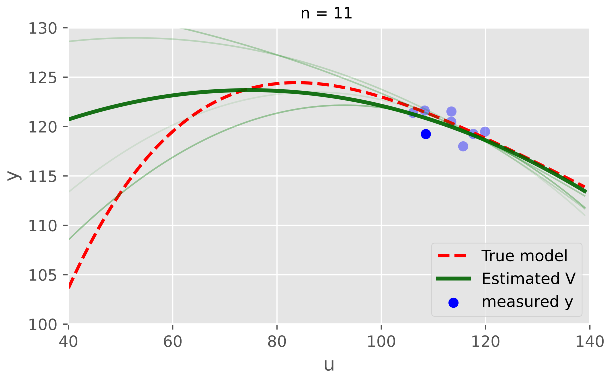


Figure 4.5: Value function estimate for a single-well system at iteration 11. Opaque colors indicate earlier measurements and estimates.

or if the measurements are subjected to noise. At every iteration, the value function weights $\boldsymbol{w}$ are computed by minimizing an objective function

$$\boldsymbol{w} = \arg\min_{w} ||y_i - \hat{y}||_2 + \lambda ||w_i||_2^2. \tag{4.8}$$

In this way, we are not learning the value function by gradual updates but instead estimating it from recent data. The regularization introduces bias but reduces variance (Russel and Norvig, 2010). The factor λ could perhaps be derived through Lipschitz analysis of the underlying model but was treated as a hyperparameter, and good values were found empirically during simulations.

As the policy moves, the local quadratic approximation must be updated to represent the true reward function better. We must therefore choose how many samples to include: $p_{\text{min}} \leq p \leq p_{\text{max}}$. Because we are updating the policy based on single samples, using more recent data will yield a better approximation at the current iterate. However, we cannot fit the curve perfectly to the data (due to an imperfect model and noise), so more samples would generally compensate for this. This becomes more apparent as the dimensionality grows; using single measurements to update the value function will give an increasing variance. Thus, there is a trade-off between variance and bias towards older data. To strike a balance, we experimented with decaying sample weights, weighting recent data more. We found no noticeable improvement, so it was removed to reduce complexity.

Another assumption we have made is that historical data exists from the producing well. We found it reasonable to let the value function be initialized by perturbing the system around the initial state according to $\boldsymbol{\theta}_0 = (\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0)$, giving some information about the input-output relations before we begin to optimize the policy.

4.2.3 SPSA

In earlier work, the SPSA gradient estimate yielded results superior to the score function gradient estimate (Grepperud, 2021). However, constrained optimization and moving optima were not explored. Therefore, we use an SPSA gradient estimate to benchmark the likelihood-ratio-based algorithms above. We implement it in the form of a stochastic Bernoulli ±1 policy, which perturbs the system twice to estimate an improvement direction for the mean.

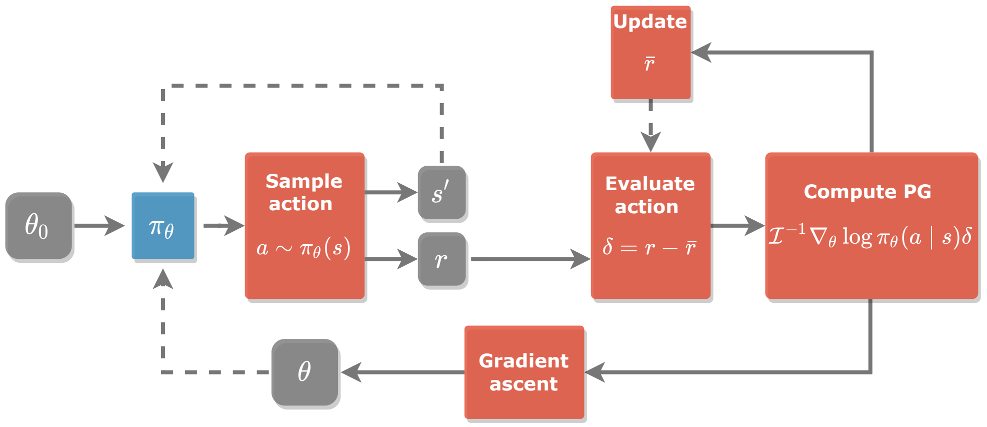
Spall (1992) proposes a scheme for decaying gain sequences based on the theoretical convergence to local optima of stochastic approximation algorithms (Robbins and Monro, 1951). For a time-horizon of N iterations, $a_{1:N}$ is implemented using SGD, while $b_{1:N}$ represents how far we perturb the system when estimating the gradient:

$$a_n = \frac{a}{(A+n)^\alpha}, \quad b_n = \frac{b}{n^\beta}. \tag{4.9}$$

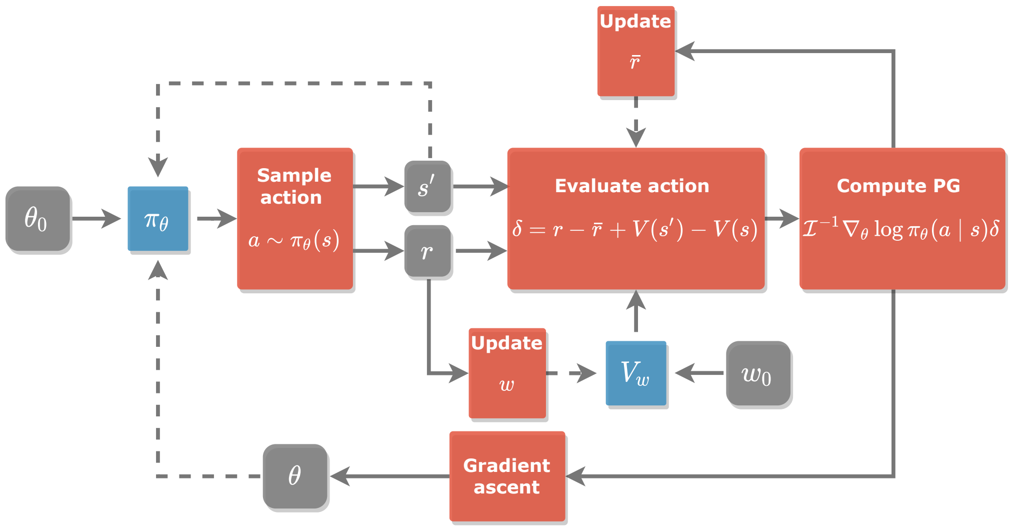
A smaller b_n would better approximate the true gradient in case of no noise. For a noisy setting, one might want to design a larger b_n. To ensure stable convergence, we set $\alpha \geq \beta$ as to ensure that b_n does not decay faster than a_n. For further details about the implementation of the algorithm and gain sequences, we refer to Spall (1998). Algorithm 3 shows the workflow of SPSA using eq. (4.9), where the policy parameter θ is the estimated mean.

Algorithm 3 SPSA with decaying gain sequences

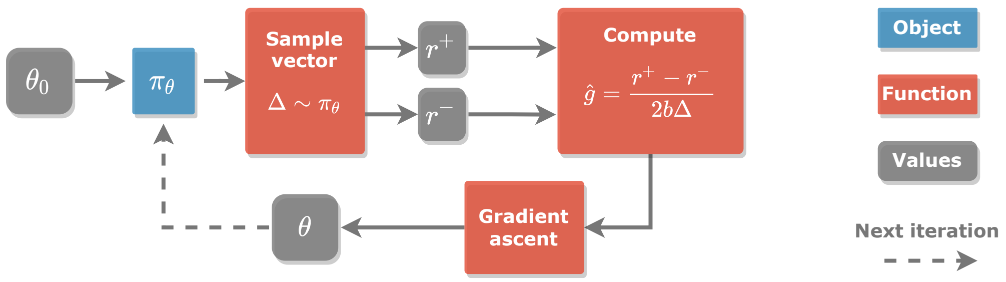
```
Require: Bernoulli $\pm 1$ policy $\pi(\theta)$, gain coefficients $a, A, \alpha, b, \beta > 0$
  for $n = 1, 2, \ldots$ do
      Set current gains $a_n, b_n$:
         $a_n = \frac{a}{(A+n)^\alpha}, \quad b_n = \frac{b}{n^\beta}$
      Sample a stochastic vector $\mathbf{\Delta}_n$:
         $\mathbf{\Delta}_n \sim \pi(\theta)$
      Perturb the system, observe rewards $r^+, r^-$:
         $r^+ = y(\theta + b_n \mathbf{\Delta}_n)$
         $r^- = y(\theta - b_n \mathbf{\Delta}_n)$
      Compute stochastic gradient:
         $\hat{g} = \frac{r^+ - r^-}{2b_n} \mathbf{\Delta}_n^{-1}$
      Update policy parameters using $\hat{g}$:
         $\theta \leftarrow \theta + a_n \hat{g}$
  end for
```



(a) Score1 with Natural gradient (Algorithm 1)



(b) Average-Reward Actor-Critic (Algorithm 2)



(c) SPSA (Algorithm 3)

Figure 4.6: Flowcharts of the stochastic algorithms

4.3 Evaluation Metrics

We employ three different metrics for measuring performance. We use *mean absolute error* (MAE) as a measure of the general performance throughout the simulation horizon of $N = 120$, corresponding to 60 days:

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N} |F_{\max,n} - Y_n|, \tag{4.10}$$

where Y_n is measured production and $F_{\max,n}$ is maximal production at time n. Minimizing MAE is analogous to maximizing overall production, as all errors are weighted equally.

As a second metric, we use Root Mean Squared Error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(F_{\max} - Y_n\right)^2}. \tag{4.11}$$

As RMSE penalizes larger errors more, it will be most affected by early iterations in the static case. A lower error will generally mean more efficient initial behavior. We use this as the primary metric for the tracking cases. Because we initialize the policies closer to the optimum, we are interested in how well the agents can track the optimum rather than general total production.

The final metric acts as a measure of distance between the initial and the optimal production levels. The value of Δ_{Yn} thus typically takes values between zero and one. We will use this distance as a way to normalize figures, making it easier to compare performance across algorithms. We use the actual production level rather than measured production due to this:

$$\Delta_{Yn} = \frac{F_{\max,n} - F_n}{F_{\max,0} - F_0} \tag{4.12a}$$

$$\Delta_{\mathrm{end}} = \frac{1}{10}\sum_{n=N-9}^{N} \Delta_{Yn} \tag{4.12b}$$

where $F_n = \sum f_i(u_{i,n})$. The value of Δ_{end} is the mean value of Δ_{Yn} over the last 10 iterations. We use this as a metric to evaluate the ability for an agent to stay at (or track) the optimum in later iterations. Thus, the three metrics emphasize the total, initial, and end performances, respectively.

Chapter 5

Simulations and Results

In this chapter, we present the simulations performed following the methodology of the last chapter. We begin with a brief implementation description and explain the overall tuning procedure. We then present the results by comparing each test case's performance and then report on algorithm-specific behavior, including some remarks on sensitivity.

The purposes of these simulations are to evaluate the applicability of minimal policy gradient optimization methods on production RTO. To do so, we have used the SPSA algorithm as a benchmark for comparing the policy gradient methods Average-Reward AC and Score1. Through the synthetic test cases, we try to capture a few phenomena that might occur on a petroleum asset, such as varying noise levels, non-static optima, and constraints. We mainly look for sample-efficiency and adaptability to these disturbances.

5.1 Implementation

All simulations have an optimization horizon of $N = 120$, corresponding to 60 days. The performance metrics are evaluated from the median data over 50 simulations. The median is used instead of the ensemble average because simulations showed that it often yielded worse performance than the median, implying that the performance distribution is negatively skewed. The models were implemented using `python3.9` (Van Rossum and Drake, 2009). We have used the `PyTorch` (Paszke et al., 2019) library for policy distributions and automatic differentiation. `scikit-learn` (Pedregosa et al., 2011) was used for regularized linear regression.

Although there are possible variations of the algorithms that are interesting to compare, we focus mainly on comparisons between Score1 with natural gradients, average-reward AC and SPSA. The simulated algorithms are referred to by the abbreviations in Table 5.1. "Score1" refers to both variations described in Section 4.2.1. To further investigate these, we have also simulated with a fixed, decaying search variance similar to SPSA.

Table 5.1: Abbreviations of algorithms

S1	Score1 with CAPG
$S1_\mu$	Score1 with CAPG, *fixed-variance*
SNG	Score1 with Natural Gradient
SNG_μ	Score1 with Natural Gradient, *fixed-variance*
AC	Average-Reward Actor-Critic
SPSA	Simultaneous Perturbation Stochastic Approximation

Table 5.2: Initial conditions

(a) Initial means: $\boldsymbol{\mu}_0$

Case	μ_1	μ_2	μ_3	μ_4	μ_5
Case 1	120	55	70	75	100
Case 2	75	65	85	-	-
Case 3	75	65	85	-	-

(b) Initial perturbation sizes: $\boldsymbol{\sigma}_0$ or b_0

Noise setting	**1D**	**3D**	**5D**
No noise	2	2	3
Low noise	2	2	3
High noise	3.5	4	6

5.1.1 Tuning Procedure and Initial Conditions

The algorithms were tuned by trial, error, and educated guesses. All parameter configurations are shown in Appendix B.1. Due to the stochasticity of the algorithms, and particularly for the Gaussian agents, we simulated each parameter configuration 50 times to find a less-variant median. We initially used 25 simulations but this still gave a too varying median. Some parameters had more influence on performance than others, but a feasible range of values for each parameter and each of the three methods was fairly easy to find. However, due to the number of parameters per algorithm and the potential for local optima when tuning one by one, we implemented a *random search* to ensure good performance. This was done by specifying a range of possible values, simulating randomly picked configurations, and comparing their median MAE. We present the results in the order that we simulated them. Therefore, we will later comment on some observations regarding the choice of parameters and tuning for the different cases and dimensions.

The initial policy means, i.e., initial inputs $\boldsymbol{u}$, are kept constant regardless of noise or dimensions. See Table 5.2a, or Figure 4.2 for a visualization with five wells. The initial perturbation magnitudes b and $\boldsymbol{\sigma}_0$ are adjusted according to the noise level – as the signal gets obscured by noise, a higher initial search variance is needed. These are shown in Table 5.2b. Thus, we treat the decaying SPSA perturbation magnitudes b_n as if they were Gaussian SDs, although this translation is not necessarily optimal. SPSA performs better with larger initial searches in 3- and 5-dimensional systems, whereas the Gaussian algorithms work better with smaller variances. We have restricted the magnitudes somewhat, according to Assumption (iii). These values are based on what we deemed as a reasonable search magnitude.

5.2 Results

The bold entries in tables indicate the best performance, i.e., the lowest MAE and RMSE, or the highest Δ_{end}. We write 1D, 3D and 5D to indicate the number of wells M. Figures for Cases 1 and 2 are normalized to take values (mostly) between 0 and 1. This is done by graphing Δ_{Yn}, which measures the distance between initial and optimal production. The distance will differ slightly for systems of different numbers of wells due to distinct production curves and initial conditions. Thus, comparing simulations done with the same dimensionality is more informative. Due to the skewed performance distribution across simulations, it is interesting to compare ensemble variances. This is done visually in some figures, where the top and bottom 20th percentiles are shown with colored bands. For figures showing inputs, these percentile bands signify the variance across simulations, not perturbation variances. We write "performance variance" when referring to the spread of the percentiles. Most figures capture the transient behavior by showing only early iterations. For Case 3, the later iterations are visualized as the constraint becomes active. We present the essential figures in this chapter. Additional figures are found in Appendix B.2.

5.2.1 Comparisons

Case 1: Static Optimization

The results of low-noise simulations are summarized in Table 5.3. The AC and SPSA algorithms achieve the lowest errors. We notice two things specifically for the 1D systems: AC performs better than the rest, and SNG outperforms S1. As the number of wells increases, SPSA performs better, achieving the lowest errors on a 5D system. At the same time, the performance gap in MAE between S1 and SNG diminishes. This can be seen from Figure 5.1. All algorithms achieve high Δ_{end}, indicating good end convergence. The SPSA performance variance increases notably in higher dimensions, despite good median performance. Figure B.1 shows this.

Simulations without noise result in a similar performance, as can be seen from Table 5.4. The SNG and AC methods improve less than SPSA, which achieves noticeably lower errors. SPSA is the only method that reaches a Δ_{end} of 1 with five wells. Figure 5.2 presents scatter plots of the median MAE of the three algorithms under varying values of σ_{noise}, corresponding to no-, low- and high-noise settings. Here, we can see how AC is the least affected by high noise in lower dimensions. The SPSA production is comparatively impaired the most, as can be read from the MAE of Table 5.5. Overall, high noise evens out performance in high dimensions. Scatter plots of RMSE under varying noise levels are shown in Figure B.2. Figures of medians and simulation variances with and without noise is presented in Figures B.3 and B.4.

Table 5.3: Case 1: Low noise

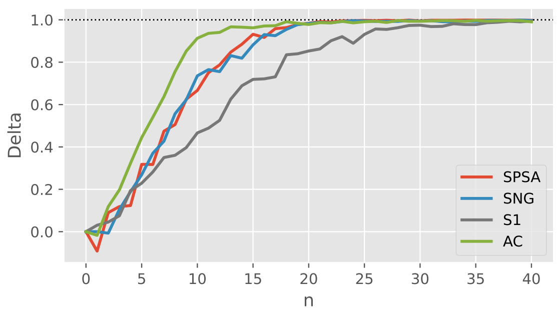
Model	**1D**			**3D**			**5D**		
	MAE	RMSE	Δ_{end}	MAE	RMSE	Δ_{end}	MAE	RMSE	Δ_{end}
S1	0.63	1.5	0.99	3.7	7.5	0.99	6.0	11.4	0.98
SNG	0.46	1.3	0.99	3.6	7.3	0.99	5.9	10.8	0.98
AC	**0.36**	**1.1**	0.99	**3.1**	**6.9**	0.99	5.6	10.8	**0.99**
SPSA	0.44	1.3	0.99	**3.1**	7.4	0.99	**5.2**	**10.4**	0.97

Table 5.4: Case 1: No noise

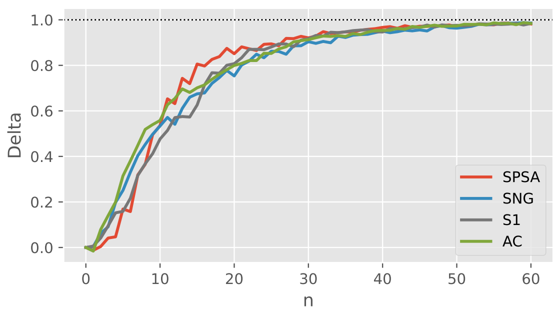
Model	**1D**			**3D**			**5D**		
	MAE	RMSE	Δ_{end}	MAE	RMSE	Δ_{end}	MAE	RMSE	Δ_{end}
SNG	0.43	1.3	1.0	3.6	7.4	1.0	5.8	10.7	0.99
AC	**0.31**	**1.0**	1.0	2.9	7.0	1.0	5.5	10.7	0.99
SPSA	0.35	1.3	1.0	**2.5**	**6.5**	1.0	**3.9**	**9.8**	**1.0**

Table 5.5: Case 1: High noise

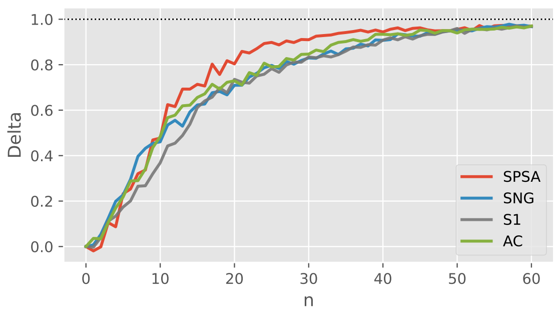
Model	**1D**			**3D**			**5D**		
	MAE	RMSE	Δ_{end}	MAE	RMSE	Δ_{end}	MAE	RMSE	Δ_{end}
SNG	0.96	1.6	**0.97**	6.4	8.4	0.84	10.8	14.0	0.88
AC	**0.74**	**1.2**	0.94	**4.9**	**7.4**	**0.93**	11.0	14.3	**0.92**
SPSA	0.98	1.7	0.89	5.6	8.4	**0.93**	**10.4**	**13.5**	**0.92**



(a) 1D

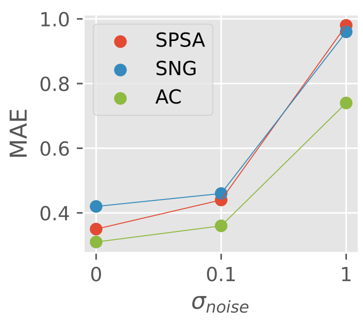


(b) 3D

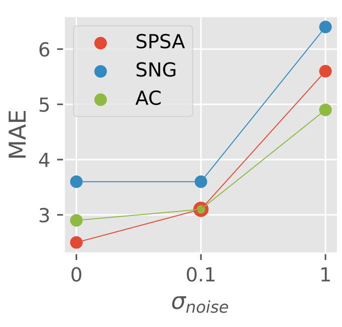


(c) 5D

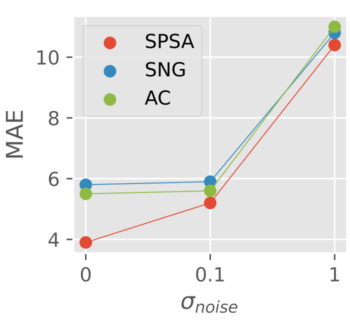
Figure 5.1: Case 1: Median Δ_{Yn} with low noise



(a) 1D



(b) 3D



(c) 5D

Figure 5.2: Case 1: Median MAE against measurement noise level σ_{noise}

Table 5.6: Case 2: Old configurations before tuning

Model	1D			3D		
	MAE	RMSE	Δ_{end}	MAE	RMSE	Δ_{end}
SNG	**0.13**	**0.15**	**0.26**	**1.2**	**1.7**	0.46
SPSA	0.22	0.31	-1.1	1.9	2.2	**0.72**

Table 5.7: Case 2: New configurations after tuning

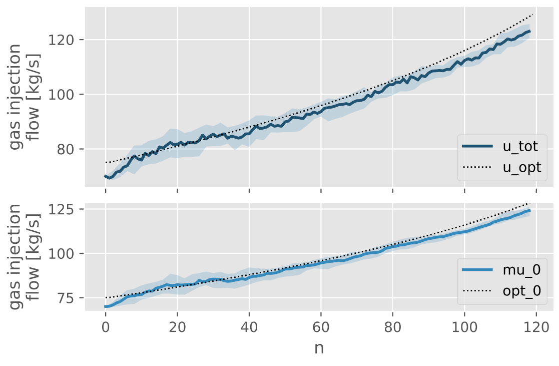
Model	1D			3D		
	MAE	RMSE	Δ_{end}	MAE	RMSE	Δ_{end}
SNG	**0.10**	**0.12**	**0.63**	**1.0**	1.4	**0.65**
SPSA	0.19	0.23	-0.51	1.0	**1.3**	0.52

Case 2: Tracking

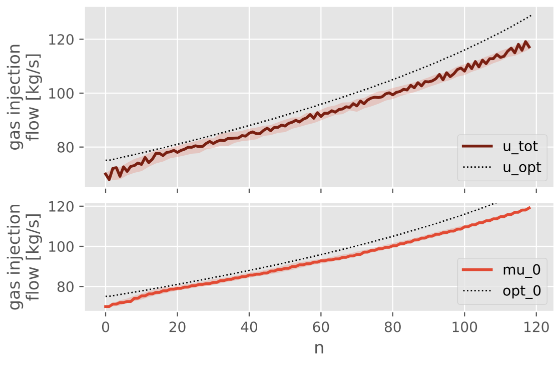
Two parameter configurations were used for the SNG and SPSA algorithms for the tracking case. Table 5.6 shows the performance of the algorithms using the parameters from the static simulations without noise. Table 5.7 shows performance after tuning to the dynamic optimum. Figure 5.3 shows how the policy means and input track the optimum for a single-well system after tuning.

Before tuning, SNG achieves about half the errors of SPSA with one well. It also finishes closer to the optimum. SPSA finishes twice as far off as the initial distance in terms of production. With three wells, SPSA finishes closer to the optimum with a higher Δ_{end}, but still tracks worse overall. See Figures 5.4a and 5.4b. The figures also shows a large variance across simulations in the 3D case, particularly following $n = 10, 30$ where the optimum abruptly moves.

SPSA improves noticeably with new parameters adapted to a dynamic optimum. Although it still fails to track as good as SNG in the 1D case, they perform on the same level in 3D. Furthermore, the performance variance is drastically reduced. As Figures 5.4c and 5.4d show, this is also the case of SNG, although the median performance improvement of SNG is scarcely noticeable. See Figure B.5 for the three-well system.

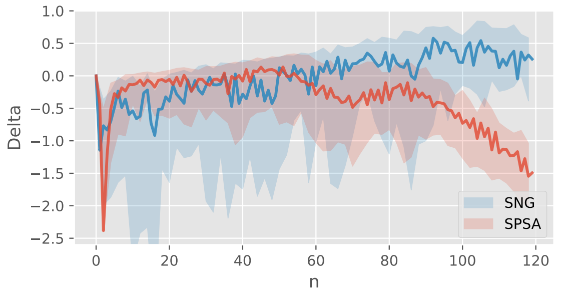


(a) 1D: SNG, new configuration

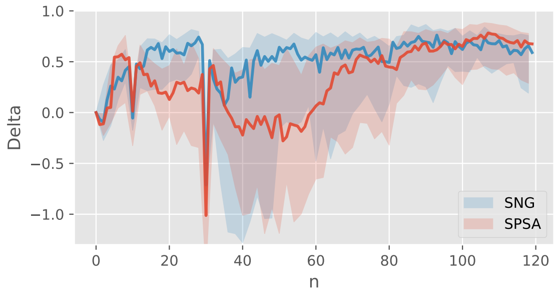


(b) 1D: SPSA, new configuration

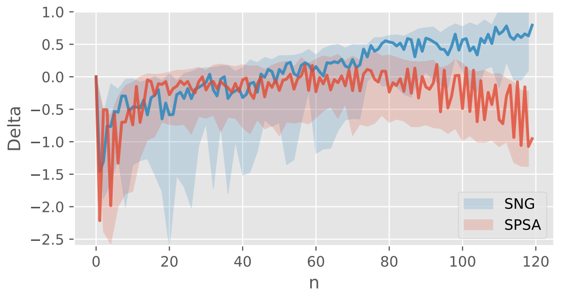
Figure 5.3: Case 2: SNG and SPSA inputs u and policy means μ (mu) for a single-well system. The optimal inputs are marked in dotted lines.



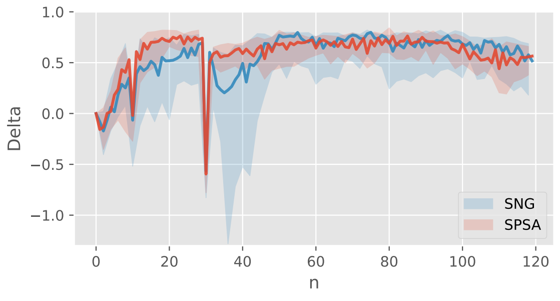
(a) 1D: Old configuration



(b) 3D: Old configuration



(c) 1D: New configuration



(d) 3D: New configuration

Figure 5.4: Case 2: Tracking performance of SNG and SPSA measured by Δ_{Yn}. In both systems, the optimum moves immediately. In the 3D system, the optimum also jumps at $n = 10, 30$.

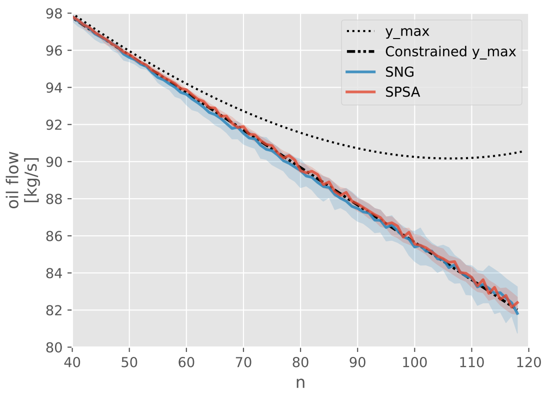
Case 3: Tracking with System-Wide Constraints

In the last test case, the SNG and SPSA algorithms track a moving, constrained optimum with $M = 1, 3$ wells. The constraint becomes active at $n = 45$ for a single-well system. At this point, the optimal production level decreases steadily while the optimal input remains at $u = 90$. The errors from the augmented single-well optimum are presented in Table 5.8. Both methods achieve errors in the same range, with SPSA performance slightly better. Figure 5.5a shows the production levels. Figures 5.6a and 5.6b show how the mean inputs adjust to the new optimum, which can be compared to the unconstrained case of Figure 5.3.

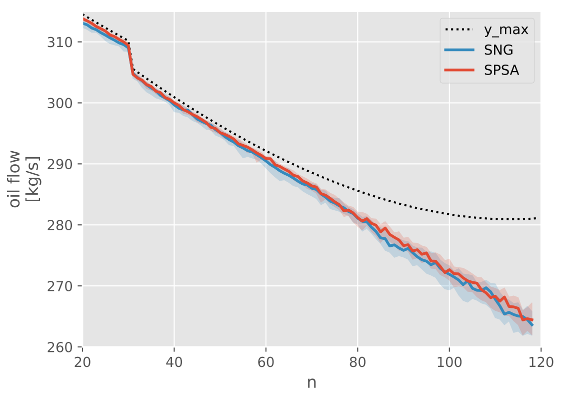
The trajectory of the three-dimensional augmented optimum is not calculated as it requires solving a non-linear constrained optimization problem at each iteration. However, Figure 5.5b allows us to inspect the behavior visually. The optimum is assumed to follow a similar, near-linear trajectory as in the 1D case. Figures 5.6c and 5.6d show policy means and the total inputs nearing the soft constraint before stabilizing. We see how SNG has a higher ensemble variance of policy means than SPSA.

Table 5.8: Case 3: Errors from augmented optimum

Model	**1D**		
	MAE	RMSE	Δ_{end}
SNG	0.14	0.18	0.18
SPSA	**0.13**	**0.16**	**0.22**

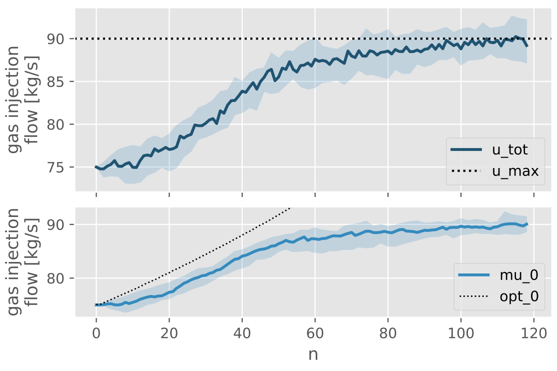


(a) 1D

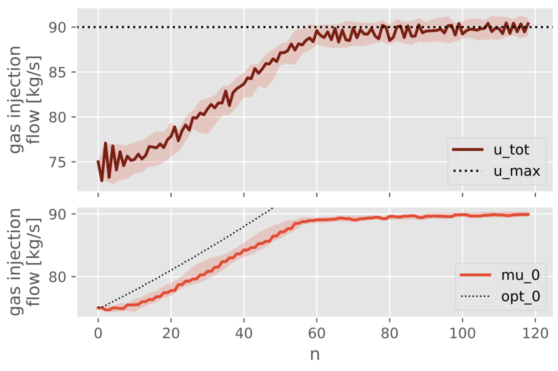


(b) 3D

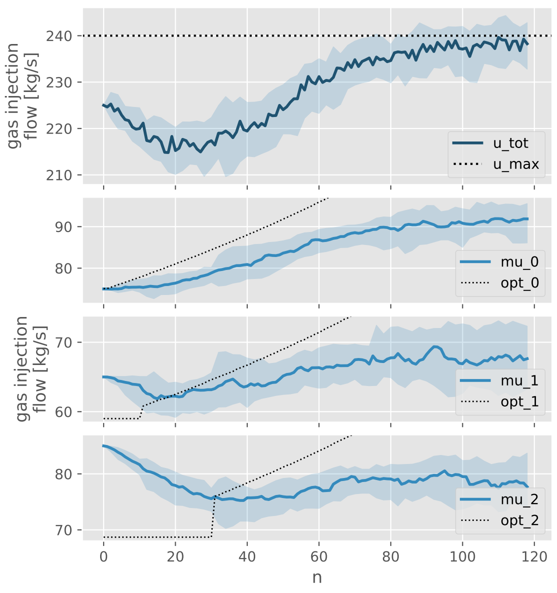
Figure 5.5: Case 3: SNG and SPSA production subjected to constraints. In the single-well system, we plot the augmented optimal production.



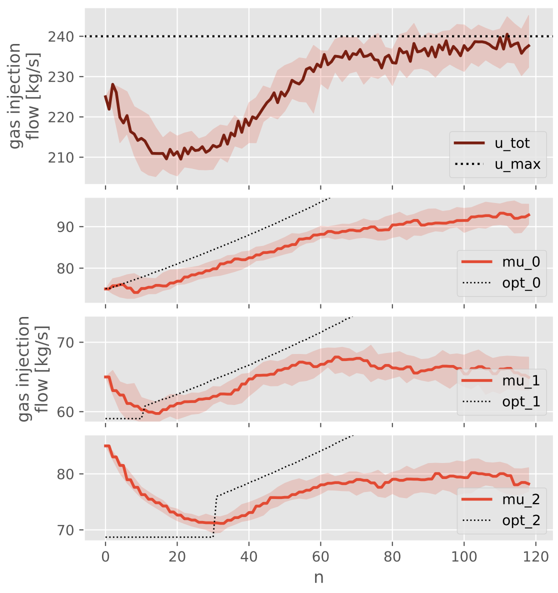
(a) 1D: SNG



(b) 1D: SPSA



(c) 3D: SNG



(d) 3D: SPSA

Figure 5.6: Case 3: SNG and SPSA inputs (u) and policy means μ (mu) with $M = 1, 3$. The unconstrained optimal inputs and $U^{\max}$ are marked in dotted lines.

5.2.2 Behavior of the Algorithms

Score1

The performance of the Score1 algorithms is mostly impeded by the high-noise setting. Simulation comparisons of SNG under varying levels of noise are presented in Figure B.6, where the low- and no-noise settings mainly differ on the five-well system. SNG surprisingly achieves the same or lower RMSE in 1D and 3D systems with low noise compared to no noise. See the scatter plots in Figure B.2 for a visual overview. The policy behavior of the low-noise simulations is shown in Figure B.12. There is a general trade-off between rapid initial convergence and more stable but slower convergence, resulting in either lower RMSE or MAE, respectively.

For the Gaussian agents, the step size of the mean α_μ influences performance the most. This is especially true for higher dimensions and increasing noise, where too high values may result in unstable policy updates. Simulating with high noise for a 5D system, the highest MAE was achieved with $\alpha_\mu = 0.04$, as opposed to $\alpha_\mu = 0.35$ without noise. High values sometimes yield low RMSE but increase the simulation variance. The variance update size α_σ also impacts the stability. We found that it was necessary to keep α_σ low to limit the possibility of exploding variances that could occur, particularly in higher dimensions. Due to this, the policy SDs in the multi-well systems remain more or less constant. The impact of the average-reward baseline can be seen from how errors increase with decreasing η in Figure B.10

To further evaluate sensitivity, we measured performance by adjusting specific parameters while keeping the rest fixed. Figure 5.7 shows the sensitivity of α_μ for S1 and SNG on a 3D system. We see how the two variants of Score1 yield patterns that are near equivalent by a scaling factor. Despite this, SNG achieves much lower errors than S1 on a 1D system. We hypothesized that this was due to the value of α_σ, which was intentionally kept very low in multi-well systems to reduce errors. The justification for this is shown in Figure B.9. By keeping $\alpha_\sigma = 10^{-3}$ in most simulations, the inverse Fisher (eq. (4.5)) works more or less as a constant. In 1D systems, a higher value for α_σ is used. Furthermore, the best performance of S1 is generally seen with policy updates that do not take the mean far out of bounds, so the clipped action policy gradient is rarely active.

We experimented with fixed-variance optimization for Score1, where the policy SDs decay according to the same sequence as the SPSA gains b_n (eq. (4.9)). This was mainly to see if the performance would increase with fewer parameters to estimate, but also to investigate the differences between S1 and SNG, as mentioned above. Table 5.9 and Figure 5.8 show how

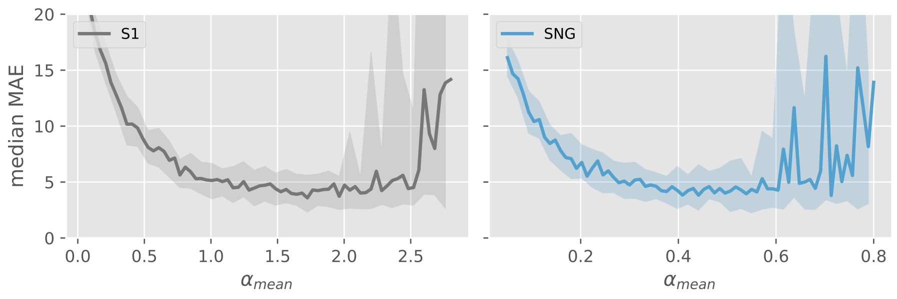
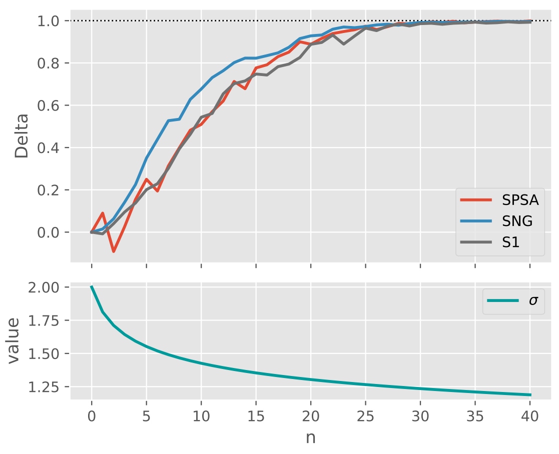


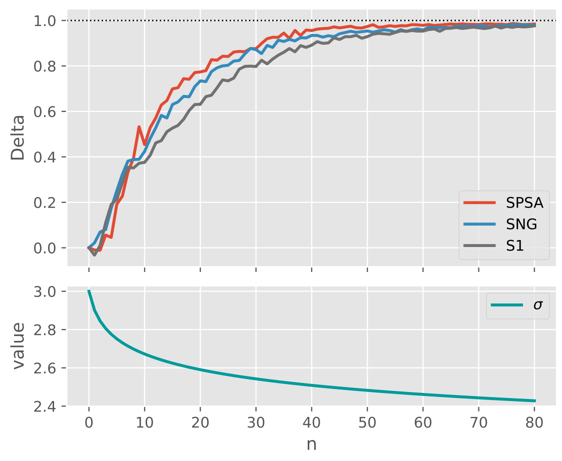
Figure 5.7: Case 1: Sensitivity of α_μ for S1 and SNG, comparing median MAE to parameter value. Different scales on x-axes.

Table 5.9: Case 1: Fixed variance optimization with low noise

Model	**1D**			**3D**			**5D**		
	MAE	RMSE	Δ_{end}	MAE	RMSE	Δ_{end}	MAE	RMSE	Δ_{end}
S1_μ	0.60	1.4	0.99	3.6	7.6	0.99	6.6	11.6	0.99
SNG_μ	**0.45**	**1.3**	0.99	3.3	7.0	**1.0**	5.5	**10.6**	0.99
SPSA	0.55	1.4	**1.0**	**2.7**	**6.4**	0.99	**5.1**	10.7	0.99



(a) 1D: $\beta = 0.15$



(b) 5D: $\beta = 0.05$

Figure 5.8: Case 1: Fixed-variance optimization with decay factor β for S1, SNG and SPSA. Low noise. Showing median Δ_{Yn} and values of σ_n, b_n

SNG outperforms S1 with decaying variances in a low noise-setting, also in higher dimensions. For performance variances and 3D simulations, see Figure B.8.

The SNG algorithm achieves a low error in the tracking case, even with parameters tuned to a static optimum. The adjustments needed for good performance are few. The average-reward weight η is increased to give a more efficient baseline. The initial SD σ_0 favors a lower value than in the static case, but a likely cause is that the initial distance to the optimum is much shorter in the tracking case. For sensitivity plots of η or $\boldsymbol{\sigma}_0$, see Figures B.10 and B.11. There is only slight improvements after tuning to the dynamic optimum, with the higher value of Δ_{end} being the most noticeable. Adding constraints to the system did not call for adjustments except a decrease in α_μ for the 1D system. In general, using a lower α_μ gives slower convergence but makes the algorithm more robust towards changes in the process model. The algorithm is sensitive to inconsistent magnitude of the TD-errors from Cases 2 and 3, as we can see from the increased performance variance after $n = 10, 30$.

Actor-Critic

The actor-critic algorithm behaves similar to SNG in most cases but tends to perform better in lower dimensions and with high noise. As with SNG, the differences between using $\sigma_{\text{noise}} = 0$ or 0.1 are slight when compared to the high-noise setting. For one- and three-well systems, AC generally achieves lower RMSE than SPSA and SNG due to the efficiency of initial iterations. See Figure B.2 for comparisons. For a performance overview of AC under varying noise levels, see Figure B.6. The policy behavior of the low-noise simulations is shown in Figure B.13.

Adding a value function introduces three new parameters that must be considered; $\lambda, p_{\min}$ and $p_{\max}$. We found that the regularization factor λ should mainly be set according to the dimension of the system. Regularization is barely needed for a 1D case with low or no noise, but we still kept it in case of low sample variance. For the multi-well systems, $\lambda \in [0.1, 0.35]$ shows good performance. Furthermore, $p_{\min}$ and $p_{\max}$ do not affect performance too much, as long as neither too few nor too many samples are included. The heuristics $p_{\min} = 2M + 5$ and $p_{\max} = 3M + 10$ seems to function across different number of wells. The other parameters were kept mostly similar to SNG, except for α_μ, which should be decreased to match the larger TD-error resulting from the addition of a value function.

SPSA

The SPSA method shows good performance overall. The algorithm converges rapidly in higher dimensions but is impacted by noise more than the other methods. The gap to the other algorithms shrinks in the case of high noise. However, this is affected to a large extent by the perturbation magnitudes b_n. For example, the initial update step sizes with 1D systems are $a = 15$ or 10, but the RMSE is the same as SNG or higher, despite SNG having more conservative step sizes. This can be seen, e.g., in Figures B.2a, B.4a and B.4b. The policy behavior of the low-noise simulations is shown in Figure B.14.

The search for parameters gave the lowest errors from using very high search variances, i.e., b_n of magnitude ± 10. However, this was deemed inconsistent with Assumption (iii). We restricted the initial search magnitude b according to Table 5.2b while adjusting the remaining coefficients. The fixed-variance optimization for the three-well case illustrates this. Comparing Table 5.3 to Table 5.9, the latter achieves a smaller error due to a larger initial search magnitude b. Figure B.8 shows how SPSA compares to Score1 with a fixed variance decay.

The tracking abilities of the SPSA algorithm are highly dependent on the gain sequences. The coefficient values giving good results in the static case benefit from decaying gains, but this decay impedes tracking performance, as seen in Figure 5.4a. The best performance is seen when drastically reducing the decay factor β. The algorithm tracks better than SNG in the 3D case but not in the 1D case. Introducing soft constraints requires higher decay rates α, β. If the values are kept as in Case 2, the mean estimate fails to converge but jumps around the optimum, inducing high average errors. SPSA achieves lower errors and ensemble variance after tuning, which can be seen from the inputs in Figure 5.6.

Chapter 6

Discussion

As presented in the previous chapter, the SPSA and AC algorithms achieve the lowest errors depending on the dimensionality and measurement noise. As they are based on two different gradient estimation principles, they inherit different characteristics, strengths, and weaknesses when considering applicability to real-time production optimization. We begin this chapter by interpreting some key findings before discussing the methods in light of the general challenges of production optimization listed in Chapter 1.

6.1 Algorithm Performances

The Score1 algorithm shows the strength of its simplicity when adapting to new scenarios. An interesting result is the performance differences between SNG and S1. The natural gradient results in better performance and seemingly accelerated learning in simulations where the policy SD varies, but not so much otherwise. This correlation is coherent with the structure of the normal Fisher information (eq. (4.5)). As the policy variance decreases over time, so will α_μ for SNG. This allows for larger initial step sizes while avoiding unstable behavior at the optimum. Thus, it counters the increasing *policy gradient* variance due to decreasing *policy* variance of the standard PG. However, this becomes more significant as σ approaches zero. Furthermore, Martens (2014) argues that smaller updates lead to better approximations of the *true* natural gradients, which are only approximated with our stochastic single-sample methods. As the coefficient values were larger in our simulations, we have likely not seen the potential of this method.

Experimenting with CAPG close to the input bounds showed its ability to direct the policy back into safe ranges. However, too large policy updates might make the policy stuck out of bounds. This happens as the log-likelihood function approaches zero when the policy mean moves further out of bounds. Step sizes were chosen to avoid this, so the performance disparity between SNG and S1 is probably caused by the natural PG. Still, the CAPG estimator seems to work better with smaller policy updates, such as those typically used with deep RL. For applications such as RTO, where samples are limited, a better way to implement input limits may be through a bounded Beta policy or just naively clipping actions as done with SNG.

Small policy updates appear to be a common factor across most practical RL implementations (e.g., Duan et al. (2016)). As we have employed a single-sample estimator, we are severely exposed to high variance, but smaller updates could conflict with Assumption (iv) and reduce efficiency. Although we did not study it for the policy gradient methods in this project, using more samples per iteration will reduce the variance of any Monte Carlo estimator (Mohamed et al., 2019). However, the trade-off between efficiency and variance persists. Earlier work (Grepperud, 2021) found that a dual-sample score-function estimator converged slower.

Two techniques that evidently work well to mitigate variance are the average-reward baseline and the value function critic. By combining these, the AC algorithm achieves lower errors, although mostly in lower dimension systems. This was expected since the multi-well critics use samples from further back in time. As the policy means shift to increase production, the local approximation is less precise and does not improve performance much.

The performance gap to SNG seems to increase with noise (except for the 5D case), suggesting that the critic compensates for higher measurement variance. This smoothing effect demonstrates how raw sample data can be combined with model estimation for faster and more stable learning. However, the critic does not track well. The curve coefficients change too rapidly for the value function to be modeled from the measurements. We did not want to impose many assumptions on the model, so we only assumed the production curves were well-behaved. We could also have enforced negative squared coefficients to ensure concavity, though this alone would probably not be enough to model the fast-moving curves.

Where AC performs the best in lower dimensions, SPSA achieves the lowest errors in higher dimensions. The SPSA algorithm is easy to tune due to the predictability of the fixed perturbation distribution. The gradient updates can be efficient with large perturbations, which are needed with high noise. Allowing arbitrarily large step sizes, SPSA achieves similar results as AC in the lower dimensions. As they apply different estimation techniques, they do not necessarily share the same optimal initial perturbation variance or other parameter configurations. However, the fixed search magnitudes lead to application-specific tuning, making SPSA seemingly less robust towards new scenarios. It generally needs more adjustments than SNG to deal with tracking and constraints. A reason for lower tracking performance could also be that the underlying cost function changes *during* the two-sided perturbation, resulting in a less accurate estimate.

The efficiency of SPSA in higher dimensions might be due to the structure of FD methods and the sampling distribution. The Bernoulli distribution can be designed to guarantee a certain perturbation magnitude. In contrast, Gaussian distributions have the probability mass centered around the mean, often leading to minor perturbations. Furthermore, the FD structure can ensure some distance between samples, which the average-reward baseline can not. The lower-variance gradient estimates compensate for the two perturbations per iteration and improve performance in higher-dimensional spaces. Buesing et al. (2016) propose an FD score function estimator that combines the two estimation techniques, giving potentially lower variance while allowing for a greater variety of distributions.

A drawback with multi-sample methods is that they do not work with general RL problems, as each sample theoretically induces a transition to a new state. For our synthetic test cases and state definition, this does not matter. With a more complex state formulation, two samples per iteration might require some adjustments to work. Furthermore, multi-sample estimates might not work if more advanced control strategies are required or there is hysteresis in the process. This brings us to whether or not it makes sense to use policy gradient methods for real-time production optimization at all.

6.1.1 Applicability to Daily Production Optimization

As we have seen, the methods are straightforward. Except for baselines and critics, policy updates happen solely from the last observations. Consequent high variance suggests that we cannot expect to find optimal policies as fast as model-based methods. Nevertheless, our interpretation is that they are relatively sample-efficient. With 120 iterations, the policy means converge to high production levels, although the optimal policy *variances* are harder to estimate within this time horizon. As a point of reference, Duan et al. (2016) benchmarks RL algorithms, including REINFORCE, using 25 million samples. Their control policies are far more complex than ours, but it is still a substantial amount.

Besides sample efficiency, the applicability of the methods depends on their adaptability and how easy they are to implement. Where the SPSA algorithm is efficient for specific cases but not as adaptable, the PG methods are general but more susceptible to high variance. This variance also makes them slightly harder to tune. However, despite that parameter values were found using random search, most values ended near our initial guesses. With proper care, the tuning should be feasible to implement online.

Based on results in the last test case, the PG methods show a potential for necessary adaptability to unmodeled disturbances. Still, we cannot trivially translate the results onto a production asset. The final test case presents various challenges, but these are hand-designed and do not nearly represent all of the limitations described in Chapter 1, nor present them accurately. Furthermore, the individual effects of some challenges, e.g., constraints, were not isolated. More advanced simulators should be used to verify the performance of these methods. In addition, the median of 50 simulations still resulted in varying median performances. The slightly lower RMSE with low noise compared to no noise Figure B.2 for SNG and AC is likely due to this. Several potential areas for future research should be explored to make policy gradient methods considerable for daily production optimization.

6.2 Further Research

A central future objective is to reduce the variance of gradient estimates. As our results suggest that a functioning baseline critic makes the PG algorithms more robust, modeling the value functions is a potential research area. Gradual weight updates (rather than discarding old data as we did), combined with higher-capacity approximators (e.g., ANNs), would allow for a more complete mapping of the environment. This could be useful as a real system is not fully decoupled, and various system-wide constraints might exist. However, time-variant dynamics imply that a global map of the reward distribution is complicated to attain.

Different policy distributions should be further investigated as they directly influence the algorithm behavior. One candidate would be the Beta distribution (see Section 3.3.3) as it naturally constrains the action space. Furthermore, it can be skewed, a property that could be exploited to direct the perturbations towards seemingly better regions. A potentially desired property of a policy could be a lower probability mass around zero to ensure a measurable effect by the perturbations. This could also lead to more noise-robust behavior. The policies could also be conditioned on a more complete state formulation, incorporating real-time measurements (such as pressures and temperatures) and other observations that affect operation.

The policy updates are also of high variance. One way to address this could be with trust-region policy updates, which restrict or penalize large updates. Although momentum SGD leads to exponentially weighted updates, a *policy network* that learns from trajectories of observations could also learn multi-stage control policies (Schulman, 2016). This would be useful for real-time dynamic control that might be required, e.g., during oscillating production caused by gas-lift (Dias et al., 2019).

Using ANNs for policies and value functions could be helpful in learning from the large amounts of accumulated historical data. Actor-critic PG methods could benefit from either being integrated with ongoing research on data-driven modeling, or learning through simulators. The general agent-environment structure allows for adapting to new situations, potentially integrating domain knowledge, and learning from monitoring current operator policies.

Chapter 7

Concluding Remarks

Breakthroughs in reinforcement learning have demonstrated the ability to find super-human control policies without relying on detailed instructions, unlike e.g., supervised learning. Silver, Singh et al. (2021) argue that a reward signal can be enough to extract *any* desired behavior from a learning agent. However, research on applications to process industries is still young. The unique challenges of a petroleum asset, which many other optimization methods do not sufficiently consider, motivate our study of policy gradient RL methods and how suitable they are for real-time production optimization.

For this purpose, we framed a gas-lift optimization problem as a reinforcement learning task, constructing a selection of test cases that simulate some of the challenges of daily production optimization. By using the score function estimator, the policy gradient algorithms find optimal, stochastic control policies for unmodeled processes. This very general estimator – commonly used in machine learning – had not yet been used for optimal control of artificial gas-lift.

Our results suggest that these methods can be as efficient as the SPSA method, which is well-researched for related optimization problems. The policy gradient algorithms show greater robustness towards different process disturbances and constraints by requiring fewer parameter adjustments. The actor-critic framework allows for efficient baselines that reduce the variance of the single-sample estimators. Critic modeling should be further researched, but we have seen how a simple linear approximator can accelerate learning with hardly any assumptions about the process. Furthermore, methods needing multiple samples per iteration are unlikely to be applied on a petroleum asset, as they often require perturbing the system in a sub-optimal direction. As control changes are of high value, algorithms based on single perturbations, such as actor-critic, could be realistic candidates for future real-world implementation.

Further research should be conducted on policy optimization, distributions, and critic design before this may be considered for real-time optimization. Still, we remain optimistic. The methods are relatively sample-efficient and adaptable, even as they are simple and easy to implement. However, our synthetic test cases are far less complex than real applications, making it difficult to conclude if the methods are suitable for real-time optimization. This thesis should be seen as a positive first step in investigating their applicability. We have demonstrated that, despite its simplicity, the underlying mathematical framework of the policy gradient methods works.

Bibliography

Amari, Shun-ichi (1998). ‘Natural Gradient Works Efficiently in Learning’. In: *MIT Press: Neural Computation 10*.

Amodei, Dario, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman and Dan Mané (2016). ‘Concrete Problems in AI Safety’. In: *CoRR*. arXiv: 1606.06565. URL: http://arxiv.org/abs/1606.06565.

Andersen, Joakim Rostrup and Lars Imsland (2021). ‘Application of Data-Driven Economic NMPC on a Gas Lifted Well Network’. In: *IFAC-PapersOnLine* 54.3. 16th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2021, pp. 275–280. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.08.254.

Bellman, Richard (1957). ‘A Markovian Decision Process’. In: *Journal of Mathematics and Mechanics*.

Berner, Christopher, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski and Susan Zhang (2019). ‘Dota 2 with Large Scale Deep Reinforcement Learning’. In: *CoRR* abs/1912.06680. arXiv: 1912.06680. URL: http://arxiv.org/abs/1912.06680.

Bieker, Hans Petter, Olav Slupphaug and Tor Johansen (Apr. 2006). ‘Real-Time Production Optimization of Offshore Oil and Gas Production Systems: A Technology Survey’. In: DOI: 10.2523/99446-MS.

Bottou, Léon, Frank E. Curtis and Jorge Nocedal (2016). *Optimization Methods for Large-Scale Machine Learning*. DOI: 10.48550/ARXIV.1606.04838. URL: https://arxiv.org/abs/1606.04838.

Buesing, Lars, Theophane Weber and Shakir Mohamed (2016). ‘Stochastic Gradient Estimation With Finite Differences’. In: *NIPS 2016: Advances in Approximate Bayesian Inference*.

Chou, Po-Wei, Daniel Maturana and Sebastian Scherer (2017). ‘Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution’. In: *Proceedings of the 34th International Conference on Machine Learning*.

Chow, Yinlam, Ofir Nachum, Aleksandra Faust, Mohammad Ghavamzadeh and Edgar A. Duéñez-Guzmán (2019). ‘Lyapunov-based Safe Policy Optimization for Continuous Control’. In: *CoRR* abs/1901.10031. arXiv: 1901.10031. URL: http://arxiv.org/abs/1901.10031.

Conn, Andrew R., Katya Scheinberg and Luis N. Vicente (2009). *Introduction to Derivative Free Optimization*. SIAM, pp. 23–31.

Corneliussen, Sidsel, Jean-Paul Couput, Eivind Dahl, Eivind Dykesteen, Kjell-Eivind Froysa, Erik Malde, Hakon Moestue, Paul Ove Moksnes, Lex Scheers and Hallvard Tunheim (2005).

Handbook of Multiphase Flow Metering. The Norwegian Society for Oil and Gas Measurement, pp. 18–25.

Dias, Ana Carolina Spindola Rangel, Felipo Rojas Soares, Johannes Jäschke, Maurício Bezerra de Souza Jr. and José Carlos Pinto (2019). 'Extracting Valuable Information from Big Data for Machine Learning Control: An Application for a Gas Lift Process'. In: *Processes*. DOI: 10.3390/pr7050252.

Do, Sy T., Fahim Forouzanfar and Albert C. Reynolds (2012). 'Estimation of Optimal Well Controls Using the Augmented Lagrangian Function with Approximate Derivatives'. In: *Proceedings of the 2012 IFAC Workshop on Automatic Control in Offshore Oil and Gas Production*.

Dochain, Denis, Michel Perrier and Martin Guay (2011). 'Extremum seeking control and its application to process and reaction systems: A survey'. In: *Mathematics and Computers in Simulation* 82.3. 6th Vienna International Conference on Mathematical Modelling, pp. 369–380. ISSN: 0378-4754. DOI: https://doi.org/10.1016/j.matcom.2010.10.022.

Duan, Yan, Xi Chen, Rein Houthooft, John Schulman and Pieter Abbeel (2016). 'Benchmarking Deep Reinforcement Learning for Continuous Control'. In: *CoRR* abs/1604.06778. arXiv: 1604.06778.

Foss, Bjarne, Brage Rugstad Knudsen and Bjarne Grimstad (2018). 'Petroleum production optimization – A static or dynamic problem?' In: *Computers & Chemical Engineering* 114. FOCAPO/CPC 2017, pp. 245–253. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2017.10.009.

Fujita, Yasuhiro and Shin-ichi Maeda (2018). *Clipped Action Policy Gradient*. DOI: 10.48550/ARXIV.1802.07564. URL: https://arxiv.org/abs/1802.07564.

Garcia, Javier and Fernando Fernandez (2015). 'A comprehensive survey on safe reinforcement learning'. In: *J. Mach. Learn. Res.* 16, pp. 1437–1480. DOI: 10.5555/2789272.2886795.

Gentle, James E., Wolfgang Härdle and Youchi Mori (2012). *Handbook of Computational Statistics. Concepts and Methods*. Springer.

Grepperud, Jakob E. (2021). 'Stochastic Optimization for Petroleum Production Assets'.

Grimstad, Bjarne, Mathilde Hotvedt, Anders T. Sandnes, Odd Kolbjørnsen and Lars Struen Imsland (2021). 'Bayesian Neural Networks for Virtual Flow Metering: An Empirical Study'. In: *CoRR* abs/2102.01391. arXiv: 2102.01391. URL: https://arxiv.org/abs/2102.01391.

Gros, Sébastien and Mario Zanon (2019). 'Data-driven Economic NMPC using Reinforcement Learning'. In: *CoRR* abs/1904.04152. arXiv: 1904.04152.

Al-Hajeri, Mubarak, M. Saeed, Jan Derks, Thomas Fuchs, T. Hantschel, Armin Kauerauf, Martin Neumaier, Oliver Schenk, Oliver Swientek and N. Tessen (June 2009). 'Basin and petroleum system modeling'. In: *Oilfield Rev.* 21.

Heger, Matthias (1994). 'Consideration of Risk in Reinforcement Learning'. In: *Machine Learning Proceedings 1994*. Ed. by William W. Cohen and Haym Hirsh. San Francisco (CA): Morgan Kaufmann, pp. 105–111. ISBN: 978-1-55860-335-6. DOI: https://doi.org/10.1016/B978-1-55860-335-6.50021-0.

Hou, Jian, Kang Zhou, Xian-Song Zhang, Xiao-Dong Kang and Hai Xie (2015). 'A review of closed-loop reservoir management'. In: *Petroleum Science 12*. DOI: 10.1007/s12182-014-0005-6.

Howard, Ronald A. (1960). 'Dynamic Programming and Markov Processes'. In: *The MIT Press*.

Howard, Ronald A. (1971). *Dynamic Probabilistic Systems. Vol II: Semi-Markov and Decision Processes*. Dover Publications.

Hu, Yujing, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu and Changjie Fan (2020). 'Learning to Utilize Shaping Rewards: A New Approach of Reward Shaping'. In: *CoRR* abs/2011.02669. arXiv: 2011.02669. URL: https://arxiv.org/abs/2011.02669.

Human Development Reports (2021). *Human Development Index*. Accessed: 2022-06-01. URL: https://hdr.undp.org/data-center/human-development-index#/indicies/HDI.

Imsland, Lars (2002). 'Output Feedback Stabilization and Control of Positive Systems'. PhD thesis. Norwegian University of Science and Technology.

International Energy Agency (2021a). 'Electricity Market Report'. In: URL: https://www.iea.org/reports/electricity-market-report-july-2021.

International Energy Agency (2021b). 'Net Zero by 2050'. In: URL: https://www.iea.org/reports/net-zero-by-2050.

Ismail, Wan Rokiah and Kukuh Trjangganung (2014). 'Mature Field Gas Lift Optimisation: Challenges & Strategies, Case Study of D-field, Malaysia'. In: *Conference Paper; International Petroleum Technology Conference, Kuala Lumpur*.

Jadid, Maharon Bin, Arne Lyngholm, Morten Opsal, Adam Vasper and Thomas White (2006). 'The Pressure's On: Innovations in Gas Lift'. In: *Oilfield Review*, pp. 44–53.

Jahanshahi, Esmaeil, Dinesh Krishnamoorthy, Andrés Codas, Bjarne Foss and Sigurd Skogestad (2019). 'Plantwide Control of an Oil Production Network'. In: *Computers and Chemical Engineering 136*.

Jäschke, Johannes and Sigurd Skogestad (2011). 'NCO tracking and self-optimizing control in the context of real-time optimization'. In: *Journal of Process Control* 21.10, pp. 1407–1416. ISSN: 0959-1524. DOI: https://doi.org/10.1016/j.jprocont.2011.07.001.

Kahneman, Daniel (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Kakade, Sham M (2001). 'A Natural Policy Gradient'. In: *Advances in Neural Information Processing Systems*. Ed. by T. Dietterich, S. Becker and Z. Ghahramani. Vol. 14. MIT Press.

Kakade, Sham M. (2003). 'On the Sample Complexity of Reinforcement Learning'. PhD thesis. University College London.

Krener, Arthur J. and Kayo Ide (2009). 'Measures of unobservability'. In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC)*, pp. 6401–6406. DOI: 10.1109/CDC.2009.5400067.

Krishnamoorthy, Dinesh, Kjetil Fjalestad and Sigurd Skogestad (2019). 'Optimal operation of oil and gas production using simple feedback control structures'. In: *Control Engineering Practice* 91, p. 104107. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2019.104107.

Lattimore, Tor and Csaba Szepesvari (2020). *Bandit Algorithms*. Cambridge University Press.

Ma, Hongze, Gaoming Yu, Yuehui She and Yongan Gu (Sept. 2019). 'Waterflooding Optimization under Geological Uncertainties by Using Deep Reinforcement Learning Algorithms'. In: *SPE Annual Technical Conference and Exhibition*. D031S043R001. DOI: 10.2118/196190-MS.

Mandziuk, J. (2007). 'Challenges for Computational Intelligence'. In: Springer. Chap. Computational Intelligence in Mind Games.

Martens, James (2014). 'New insights and perspectives on the natural gradient method'. In: *Journal of Machine Learning Research 21*. DOI: 10.48550/ARXIV.1412.1193.

Miftakhov, Ruslan, Abdulaziz Al-Qasim and Igor Efremov (Jan. 2020). 'Deep Reinforcement Learning: Reservoir Optimization from Pixels'. In: *IPTC*. DOI: 10.2523/IPTC-20151-MS.

Mnih, Volodymyr, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver and Koray Kavukcuoglu (2016). 'Asynchronous Methods for Deep Reinforcement Learning'. In: *CoRR* abs/1602.01783. arXiv: 1602.01783. URL: http://arxiv.org/abs/1602.01783.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra and Martin A. Riedmiller (2013). 'Playing Atari with Deep Reinforcement Learning'. In: *CoRR* abs/1312.5602. arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

Mohamed, Shakir, Mihaela Rosca, Michael Figurnov and Andriy Mnih (2019). 'Monte Carlo Gradient Estimation in Machine Learning'. In: DOI: 10.48550/ARXIV.1906.10652. URL: https://arxiv.org/abs/1906.10652.

Mohammadpoor, Mehdi and Farshid Torabi (Dec. 2018). 'Big Data analytics in oil and gas industry: An emerging trend'. In: *Petroleum* 6. DOI: 10.1016/j.petlm.2018.11.001.

Morari, Manfred and Jay H. Lee (1999). 'Model predictive control: past, present and future'. In: *Computers & Chemical Engineering* 23.4, pp. 667–682. ISSN: 0098-1354. DOI: https://doi.org/10.1016/S0098-1354(98)00301-9.

Nocedal, Jorge and Stephen J. Wright (2006). *Numerical Optimization*. Springer.

OpenAI (2018). *Spinning Up - Vanilla Policy Gradient*. https://spinningup.openai.com/en/latest/algorithms/vpg.html. [Online; accessed 20-May-2022].

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba and Lei Zhang (2019). 'Solving Rubik's Cube with a Robot Hand'. In: *CoRR* abs/1910.07113. arXiv: 1910.07113. URL: http://arxiv.org/abs/1910.07113.

Palen, W. and A. Goodwin (1996). 'Increasing Production in a Mature Basin: "The Choke Model"'. In: *SPE Europec featured at EAGE Conference and Exhibition* All Days. SPE-36848-MS. DOI: 10.2118/36848-MS. URL: https://doi.org/10.2118/36848-MS.

Pan, Elton, Panagiotis Petsagkourakis, Max Mowbray, Dongda Zhang and Ehecatl Antonio del Rio-Chanona (2021). 'Constrained model-free reinforcement learning for process optimization'. In: *Computers & Chemical Engineering* 154, p. 107462. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2021.107462.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai and Soumith Chintala (2019). 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pavlov, Alexey, Mark Haring and Kjetil Fjalestad (Dec. 2017). 'Practical extremum-seeking control for gas-lifted oil production'. In: pp. 2102–2107. DOI: 10.1109/CDC.2017.8263957.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg et al. (2011). 'Scikit-learn: Machine learning in Python'. In: *Journal of machine learning research* 12.Oct, pp. 2825–2830.

Peters, Jan and Stefan Schaal (Nov. 2006). 'Policy Gradient Methods for Robotics'. In: vol. 2006, pp. 2219–2225. DOI: 10.1109/IROS.2006.282564.

Petrazzini, Irving and Eric Antonelo (2021). ‘Proximal Policy Optimization with Continuous Bounded Action Space via the Beta Distribution’. In: *arXiv:2111.02202v1 [cs.LG]*.

Petsagkourakis, Panagiotis, Ilya Orson Sandoval, Eric Bradford, Federico Galvanin, Dongda Zhang and Ehecatl Antonio del Rio-Chanona (2020). ‘Chance Constrained Policy Optimization for Process Control and Optimization’. In: DOI: 10.48550/ARXIV.2008.00030.

Rashid, Kashif, William Bailey and B. Couet (June 2012). ‘A Survey of Methods for Gas-Lift Optimization’. In: *Modelling and Simulation in Engineering* 2012. DOI: 10.1155/2012/516807.

Robbins, Herbert and Sutton Monro (1951). ‘A Stochastic Approximation Method’. In: *The Annals of Mathematical Statistics*.

Russel, Stuart and Peter Norvig (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Sadegh, P. and J. Spall (1998). ‘Optimal random perturbations for stochastic approximation using a simultaneous perturbation gradient approximation’. In: *IEEE Transactions on Automatic Control* 43.10, pp. 1480–1484. DOI: 10.1109/9.720513.

Schulman, John (2016). ‘Optimizing Expectations: From Deep Reinforcement Learning to Stochastic Computational Graphs’. PhD thesis. Berkeley, University of California.

Schulman, John, Sergey Levine, Philipp Moritz, Michael Jordan and Pieter Abbeel (2015). ‘Trust Region Policy Optimiization’. In: *Proceedings of the 31st International Conference on Machine Learning*.

Schulman, John, Philipp Moritz, Sergey Levine, Michael Jordan and Pieter Abbeel (2016). ‘High-Dimensional Continuous Control Using Generalized Advantage Estimation’. In: *ICLR Conference Paper*.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford and Oleg Klimov (2017). ‘Proximal Policy Optimization Algorithms’. In: *arXiv:1707.06347 [cs.LG]*.

Shu-Jun Liu, Miroslav Krstic (2012). *Stochastic Averaging and Stochastic Extremum Seeking*. Springer London, pp. 11–17. DOI: https://doi.org/10.1007/978-1-4471-4087-0.

Silva, Thiago and Alexey Pavlov (2020). ‘Dither signals optimization in constrained multi-agent extremum seeking control’. In: *IFAC Papers*.

Silver, David (2015). *Lectures on Reinforcement Learning*. URL: https://www.davidsilver.uk/teaching/.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel and Demis Hassabis (2016). ‘Mastering the game of Go with deep neural networks and tree search’. In: *Nature*.

Silver, David, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra and Martin Riedmiller (2014). ‘Deterministic Policy Gradient Algorithms’. In: *Proceedings of the 31 st International Conference on Machine Learning*.

Silver, David, Satinder Singh, Doina Precup and Richard S. Sutton (2021). ‘Reward is enough’. In: *Artificial Intelligence* 299. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2021.103535.

Sircar, Anirbid, Kriti Yadav, Kamakshi Rayavarapu, Namrata Bist and Hemangi Oza (2021). ‘Application of machine learning and artificial intelligence in oil and gas industry’. In: *Petroleum Research* 6.4, pp. 379–391. ISSN: 2096-2495. DOI: https://doi.org/10.1016/j.ptlrs.2021.05.009.

Spall, J. (1992). 'Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation'. In: *IEEE Transactions on Automatic Control*.

Spall, J. (1997). 'A One-measurement Form of Simultaneous Stochastic Approximation'. In: *Automatica*.

Spall, J. (1998). 'Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization'. In: *IEEE Transactions on Aerospace and Electronic Systems*.

Spall, J. (2012). 'Handbook of Computational Statistics'. In: ed. by James E. Gentle, Wolfgang Härdle and Youchi Mori. Springer. Chap. 7.

Sutton, Richard S. and Andrew G Barto (2020). *Reinforcement Learning: An Introduction*. Second. MIT press.

Sutton, Richard S., David McAllester, Satinder Singh and Yishay Mansour (1999). 'Policy Gradient Methods for Reinforcement Learning with Function Approximation'. In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*.

Van Rossum, Guido and Fred L. Drake (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace. ISBN: 1441412697.

Production Optimization in Closed-Loop Reservoir Management (Nov. 2007). Vol. All Days. SPE Annual Technical Conference and Exhibition. SPE-109805-MS. DOI: 10.2118/109805-MS. URL: https://doi.org/10.2118/109805-MS.

Wasserman, Larry (2013). *All of Statistics: A Concise Course*. Springer, pp. 126–134.

Watkins, Christopher (1989). 'Learning from Delayed Rewards'. PhD thesis. King's College.

Wilks, S. S. (1962). *Mathematical Statistics*. J. Wiley and Sons.

Williams, Ronald J. (1992). 'Simple statistical gradient-following algorithms for connectionist reinforcement learning'. In: *Mach Learn*.

Wu, Yuhuai, Elman Mansimov, Shun Liao, Roger Grosse and Jimmy Ba (2017). 'Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation'. In: *arXiv:1708.05144v2 [cs.LG]*.

Appendix A

Derivations

A.1 Derivation of the Score Function Gradient Estimator

The score function is given by

$$\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x} \mid \boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}} p(\mathbf{x} \mid \boldsymbol{\theta})}{p(\mathbf{x} \mid \boldsymbol{\theta})}. \tag{A.1}$$

By manipulating the expression for the expected value of f, and inserting the score function, we can find an unbiased estimator for the stochastic gradient $\boldsymbol{g}$:

$$\begin{aligned}
\boldsymbol{g} &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{x \sim p(\boldsymbol{\theta})}[f(x)] = \nabla_{\boldsymbol{\theta}} \int p(x \mid \boldsymbol{\theta}) f(x) dx && \text{(A.2a)} \\
&= \int f(x) \nabla_{\boldsymbol{\theta}} p(x \mid \boldsymbol{\theta}) dx && \text{(A.2b)} \\
&= \int p(x \mid \boldsymbol{\theta}) f(x) \nabla_{\boldsymbol{\theta}} \log p(x \mid \boldsymbol{\theta}) dx && \text{(A.2c)} \\
&= \mathbb{E}_{x \sim p(\boldsymbol{\theta})}\big[f(x) \nabla_{\boldsymbol{\theta}} \log p(x \mid \boldsymbol{\theta})\big] && \text{(A.2d)}
\end{aligned}$$

This derivation is valid when the the measure $p_\theta(\boldsymbol{x}) = p(\boldsymbol{x} \mid \boldsymbol{\theta})$ satisfies three conditions:

1. $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ must be continuously differentiable in $\boldsymbol{\theta}$
2. $f(\boldsymbol{x}) p(\boldsymbol{x} \mid \boldsymbol{\theta})$ is differentiable and integrable for all $\boldsymbol{\theta}$
3. $\sup_{\boldsymbol{\theta}} \|f(\mathbf{x}) \nabla_{\boldsymbol{\theta}} p(\mathbf{x} \mid \boldsymbol{\theta})\|_1 \leq g(\mathbf{x}) \forall \mathbf{x}$ for some integrable function $g(\boldsymbol{x})$

An implicit assumption is also made on absolute continuity, meaning that $p(\boldsymbol{x} \mid \boldsymbol{\theta} + h) > 0$ where $p(\boldsymbol{x} \mid \boldsymbol{\theta}) > 0$. This holds for most distributions of interest. A counter-example is the uniform distribution, since $\boldsymbol{\theta}$ defines its support. This can lead to a biased gradient estimate. We refer to Mohamed et al. (2019) for more details.

A.2 The Gaussian Score Function

Consider a stochastic variable $\boldsymbol{X} = (X_1, ..., X_n)^\top$ where each component X_i is normally distributed, so that the vector follows a multivariate Gaussian distribution, $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The probability density function of $\boldsymbol{X}$ can be written out as

$$p_X(x_1, ..., x_n; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}, \tag{A.3}$$

with $|\boldsymbol{\Sigma}|$ representing the determinant of $\boldsymbol{\Sigma}$. If we let every pair of variables (X_i, X_j) be independent, the multivariate distribution simplifies to a product of univariate distributions; a diagonal Gaussian distribution. The determinant of the covariance matrix $\boldsymbol{\Sigma}$ is the product of the diagonal entries. To derive the score function with respect to the parameters $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we use the log-likelihood:

$$\begin{aligned}
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) && \text{(A.4a)}\\
&= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\Big(\prod_{i=1}^{n}\sigma_i^2\Big) - \frac{1}{2}\sum_{i=1}^{n}\frac{(x_i-\mu_i)^2}{\sigma_i^2} && \text{(A.4b)}\\
&= \text{const} - \frac{1}{2}\sum_{i=1}^{n}\left\{\log(\sigma_i^2) + \frac{(x_i-\mu_i)^2}{\sigma_i^2}\right\} && \text{(A.4c)}
\end{aligned}$$

The derivatives of the log-likelihood with respect to $\boldsymbol{\theta}$ are:

$$\begin{aligned}
\frac{\partial}{\partial \mu_i}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \frac{x_i-\mu_i}{\sigma_i}, && \text{(A.5a)}\\
\frac{\partial}{\partial \sigma_i}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= -\frac{1}{2}\left(\frac{2}{\sigma_i} - \frac{2(x_i-\mu_i)^2}{\sigma_i^3}\right) && \text{(A.5b)}\\
&= \frac{1}{\sigma_i}\left(\frac{(x_i-\mu_i)^2}{\sigma_i^2} - 1\right) && \text{(A.5c)}
\end{aligned}$$

This gives us the diagonal Gaussian score function:

$$\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \begin{bmatrix} \frac{x_1-\mu_1}{\sigma_1^2} & \frac{1}{\sigma_1}\left(\frac{(x_1-\mu_1)^2}{\sigma_1^2} - 1\right) \\ \vdots & \vdots \\ \frac{x_n-\mu_n}{\sigma_n^2} & \frac{1}{\sigma_n}\left(\frac{(x_n-\mu_n)^2}{\sigma_n^2} - 1\right) \end{bmatrix}. \tag{A.6}$$

A.2.1 Variance Map

In a discretized system, one might want to ensure positive-definiteness of the diagonal covariance matrix. This can be done e.g., by using an exponential mapping $\sigma = e^s + c$, with $c \geq 0$ as a stability constant. We derive a new expression for the score function:

$$\begin{aligned}
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\Big(\prod_{i=1}^{n}(e^{s_i}+c)^2\Big) - \frac{1}{2}\sum_{i=1}^{n}\frac{(x_i-\mu_i)^2}{(e^{s_i}+c)^2} && \text{(A.7a)}\\
&= \text{const} - \frac{1}{2}\sum_{i=1}^{n}\left\{2\log(e^{s_i}+c) + \frac{(x_i-\mu_i)^2}{(e^{s_i}+c)^2}\right\} && \text{(A.7b)}
\end{aligned}$$

Differentiating with respect to the new parameters $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{s})$, with a diagonal covariance matrix gives us that

$$\frac{\partial}{\partial \mu_i} \log p_\theta(\boldsymbol{x}) = \frac{x_i - \mu_i}{(e^{s_i} + c)^2}, \tag{A.8a}$$

$$\frac{\partial}{\partial s_i} \log p_\theta(\boldsymbol{x}) = -\frac{e^{s_i}}{e^{s_i} + c} + \frac{2e^{s_i}(x_i - \mu_i)^2}{(e^{s_i} + c)^3} \tag{A.8b}$$

$$= \frac{e^{s_i}}{e^{s_i} + c}\left(\frac{2(x_i - \mu_i)^2}{(e^{s_i} + c)^2} - 1\right), \tag{A.8c}$$

which replace the expressions of eq. (A.6). For other mappings of the variance, the same procedure is followed to derive the gradient expressions.

A.2.2 Behaviour of the Score Function

These gradients can be used to update the distributional parameters $\boldsymbol{\theta}$. We can see that the gradient component for μ_i is negative when

$$\frac{x_i - \mu_i}{\sigma_i^2} < 0$$
$$x_i - \mu_i < 0,$$

meaning that the mean is shifted towards the variable x_i and is scaled with the variance. Likewise, the gradient component for the variance σ_i^2 is negative when

$$\frac{1}{\sigma_i}\left(\frac{(x_i - \mu_i)^2}{\sigma_i^2} - 1\right) < 0$$
$$(x_i - \mu_i)^2 < \sigma_i^2$$
$$\left|x_i - \mu_i\right| < \sigma_i.$$

We thus have that the standard deviation σ_i will be decreased if the Euclidean distance between μ_i and x_i is lower than the standard deviation σ_i, and increases in the opposite case, inversely proportional to the size of σ_i^2. This holds true regardless of any mapping applied to σ^2, including *softplus*, *softmax*, or *exponential*, which can be seen if inserting the mapping into the above inequalities.

For a policy gradient update (see Section 3.2.3), the gradient is multiplied with a feedback-signal (e.g. R, δ), so the sign might change, depending on the reward. For instance, if a sample is far from the mean, and gives a positive reward change, the variance is increased. If a sample closer to the mean yields positive feedback, variance is decreased.

A.3 Fisher Information Matrix

The Fisher information matrix (Sham M Kakade, 2001) of a distribution $p(x \mid \boldsymbol{\theta})$ is defined as:

$$\mathcal{I}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim p(\boldsymbol{\theta})}\left[\frac{\partial \log p(x \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i}\frac{\partial \log p(x \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}_j}\right] \tag{A.9a}$$

$$= \mathbb{E}_{x\sim p(\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}} \log p(x \mid \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(x \mid \boldsymbol{\theta})^{\top}\right], \tag{A.9b}$$

which can be approximated by a Monte Carlo estimator.

For a Gaussian distribution, we can derive the expressions for the derivatives of the log-likelihood $l(\boldsymbol{\theta}) = \log p(x \mid \boldsymbol{\theta})$ analytically, as done in Chou et al. (2017):

$$\mathcal{I}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim p(\boldsymbol{\theta})}\begin{bmatrix} \frac{\partial^2 l}{\partial \mu^2} & \frac{\partial^2 l}{\partial \mu \partial \sigma} \\ \frac{\partial^2 l}{\partial \mu \partial \sigma} & \frac{\partial^2 l}{\partial \sigma^2} \end{bmatrix} \tag{A.10a}$$

$$= \mathbb{E}_{x\sim p(\boldsymbol{\theta})}\begin{bmatrix} -\frac{1}{\sigma^2} & \frac{-2(x-\mu)}{\sigma^3} \\ \frac{-2(x-\mu)}{\sigma^3} & \frac{-3(x-\mu)^2}{\sigma^4} + \frac{1}{\sigma^2} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{2}{\sigma^2} \end{bmatrix} \tag{A.10b}$$

This holds under certain regularity conditions, that are essentially just smoothness conditions on $p(x \mid \boldsymbol{\theta})$ (Wasserman, 2013).

Appendix B

Simulations

B.1 Parameter Configurations

Unless stated otherwise, the simulations are for the static Case 1 (Section 5.2.1). The asterisk (*) indicates a fixed standard deviation decay factor β.

Table B.1: Score1 with Clipped Action Policy Gradient (S1)

	Simulation	References	α_μ	α_σ	γ	η
1D	Low noise	Tab 5.3, Fig 5.1a	3.6	0.1	0.5	0.7
	Fixed-variance	Tab 5.9, Fig B.8a	3.6	0.15*	0.6	0.6
3D	Low noise	Tab 5.3, Fig 5.1b	1.7	0.01	0.6	0.6
	Fixed-variance	Tab 5.9, Fig B.8c	1.6	0.10*	0.5	0.6
5D	Low noise	Tab 5.3, Fig 5.1c	1.3	0.001	0.4	0.6
	Fixed-variance	Tab 5.9, Fig B.8e	0.25	0.05*	0.5	0.6

Table B.2: Score1 with Natural Gradient (SNG)

	Simulation	References	α_μ	α_σ	γ	η
1D	No noise	Tab 5.4, Fig B.6a	1.2	0.05	0.6	0.7
	Low noise	Tab 5.3, Fig B.12a	1.	0.05	0.6	0.7
	High noise	Tab 5.5, Fig B.6a	0.2	0.01	0.6	0.7
	Fixed-variance	Tab 5.9, Fig B.8a	2.2	0.15*	0.6	0.6
	Tracking	Tab 5.7, Fig 5.4c	1.	0.01	0.7	0.95
	Constrained	Tab 5.8, Fig 5.6a	0.7	0.01	0.7	0.95
3D	No noise	Tab 5.4, Fig B.6c	0.55	0.001	0.5	0.6
	Low noise	Tab 5.3, Fig B.12b	0.5	0.001	0.5	0.6
	High noise	Tab 5.5, Fig B.6c	0.13	0.001	0.5	0.6
	Fixed-variance	Tab 5.9, Fig B.8c	0.3	0.10*	0.5	0.6
	Tracking	Tab 5.7, Fig 5.4d	0.45	0.001	0.6	0.9
	Constrained	Tab 5.8, Fig 5.6c	0.45	0.001	0.6	0.9
5D	No noise	Tab 5.4, Fig B.6e	0.35	0.001	0.3	0.6
	Low noise	Tab 5.3, Fig B.12c	0.25	0.001	0.3	0.6
	High noise	Tab 5.5, Fig B.6e	0.04	0.001	0.4	0.6
	Fixed-variance	Tab 5.9, Fig B.8e	0.25	0.05*	0.5	0.6

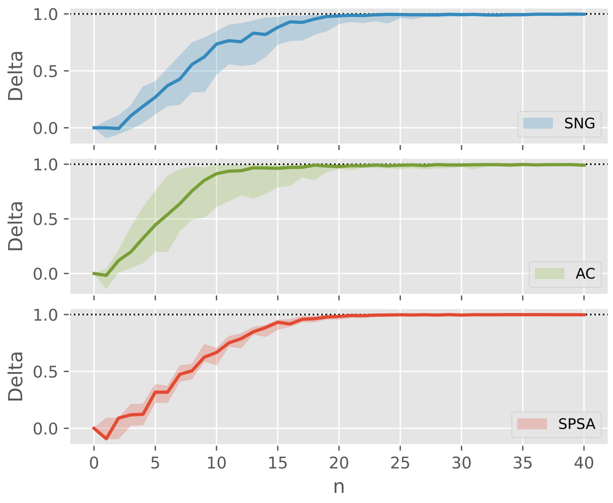
Table B.3: Actor-Critic (AC)

	Simulation	**References**	α_μ	α_σ	γ	η	$p_{\min}$	$p_{\max}$	λ
	No noise	Tab 5.4, Fig B.6b	1.2	0.05	0.5	0.7	7	13	0.02
1D	Low noise	Tab 5.3, Fig B.13a	1.	0.01	0.5	0.7	7	13	0.02
	High noise	Tab 5.5, Fig B.6b	0.2	0.01	0.6	0.7	7	13	0.05
	No noise	Tab 5.4, Fig B.6d	0.35	0.01	0.4	0.6	11	19	0.1
3D	Low noise	Tab 5.3, Fig B.13b	0.25	0.001	0.4	0.6	11	19	0.1
	High noise	Tab 5.5, Fig B.6d	0.05	0.001	0.4	0.6	11	19	0.2
	No noise	Tab 5.4, Fig B.6f	0.2	0.001	0.4	0.5	15	25	0.2
5D	Low noise	Tab 5.3, Fig B.13c	0.12	0.001	0.4	0.4	15	25	0.2
	High noise	Tab 5.5, Fig B.6f	0.015	0.001	0.4	0.4	15	25	0.3

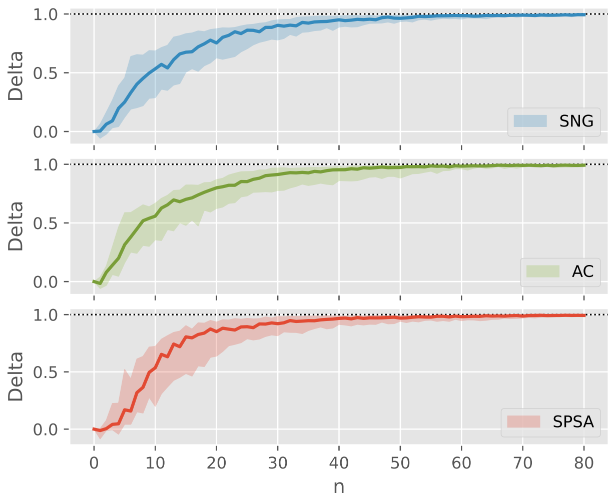
Table B.4: Simultaneous Perturbation Stochastic Approximation (SPSA)

	Simulation	**References**	a	A	α	b	β	γ
	No noise	Tab 5.4, Fig B.7a	15	1	0.3	2	0.3	0.5
	Low noise	Tab 5.3, Fig B.14a	15	1	0.4	2	0.2	0.6
1D	High noise	Tab 5.5, Fig B.7a	9	1	0.4	3.5	0.2	0.5
	Tracking	Tab 5.7, Fig 5.4c	7	1	0.05	2	0.05	0.5
	Constrained	Tab 5.8, Fig 5.6b	5	1	0.2	2	0.2	0.5
	No noise	Tab 5.4, Fig B.3c	12	1	0.2	2	0.2	0.5
	Low noise	Tab 5.3, Fig B.14b	10	6	0.2	2	0.2	0.5
3D	High noise	Tab 5.5, Fig B.3d	5	3	0.3	4	0.1	0.5
	Tracking	Tab 5.7, Fig 5.4d	5	1	0.05	2	0.01	0.4
	Constrained	Tab 5.8, Fig 5.6d	5	1	0.1	2	0.1	0.3
	No noise	Tab 5.4, Fig B.7b	10	3	0.2	2	0.2	0.5
5D	Low noise	Tab 5.3, Fig B.14c	10	6	0.2	3	0.2	0.3
	High noise	Tab 5.5, Fig B.7b	5	3	0.3	6	0.1	0.5

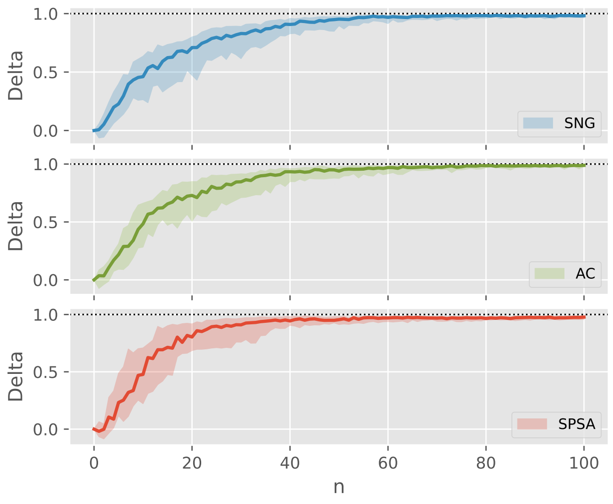
B.2 Supplementary Figures



(a) 1D: 80-20 percentiles

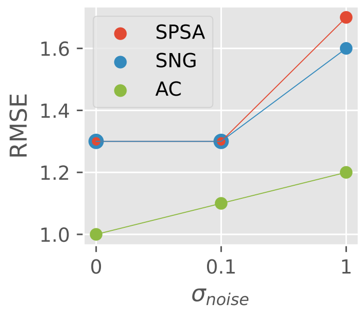


(b) 3D: 80-20 percentiles

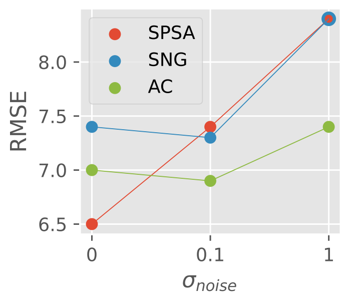


(c) 5D: 80-20 percentiles

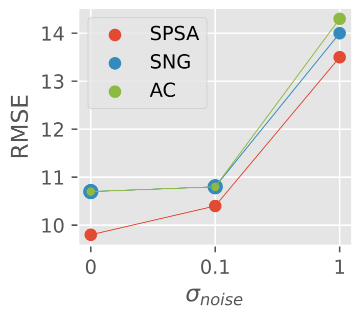
Figure B.1: Case 1: Comparisons of SNG, AC and SPSA algorithms with low noise.



(a) 1D

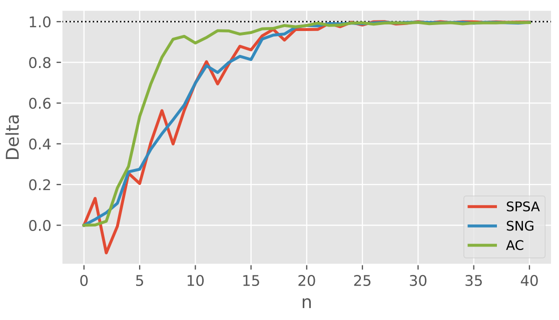


(b) 3D

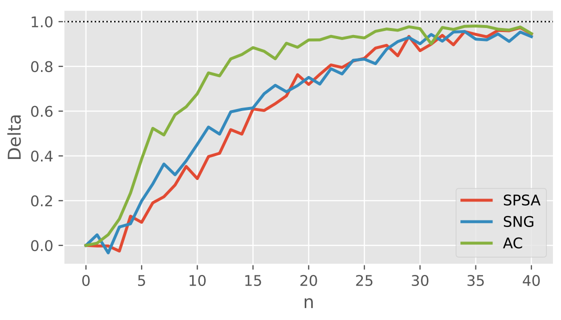


(c) 5D

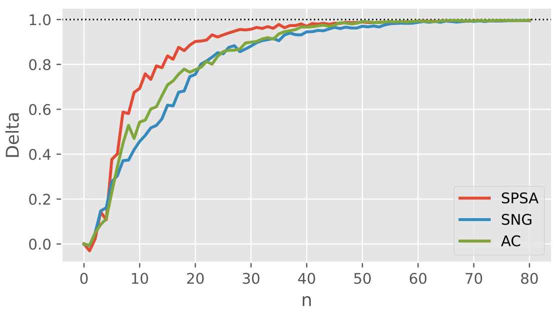
Figure B.2: Case 1: Median RMSE against measurement noise level σ_{noise}



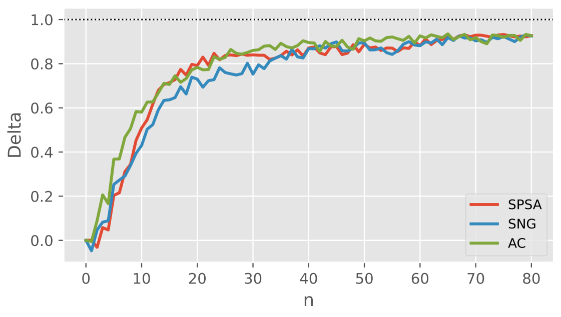
(a) 1D: No noise



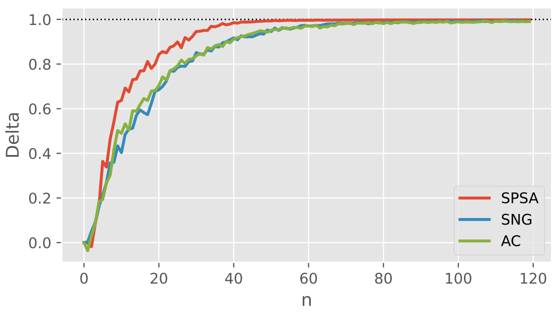
(b) 1D: High noise



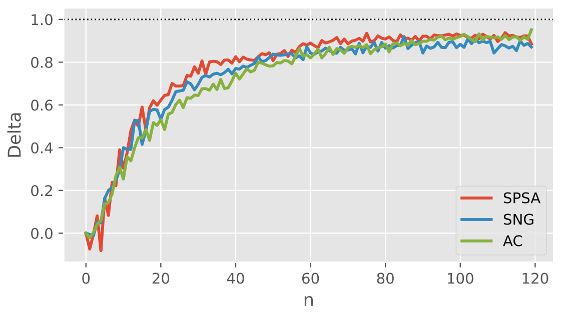
(c) 3D: No noise



(d) 3D: High noise

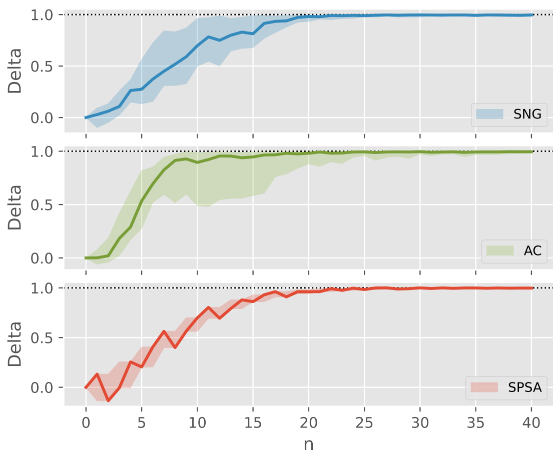


(e) 5D: No noise

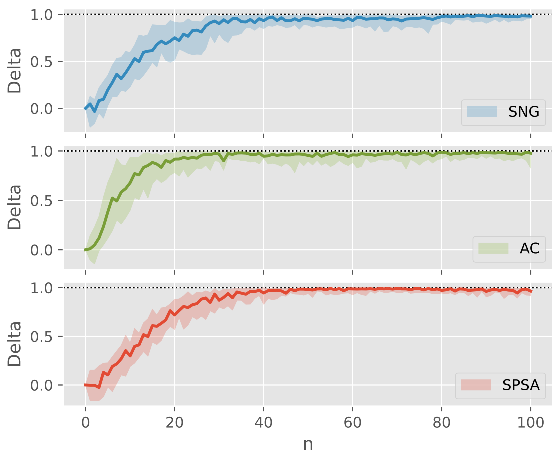


(f) 5D: High noise

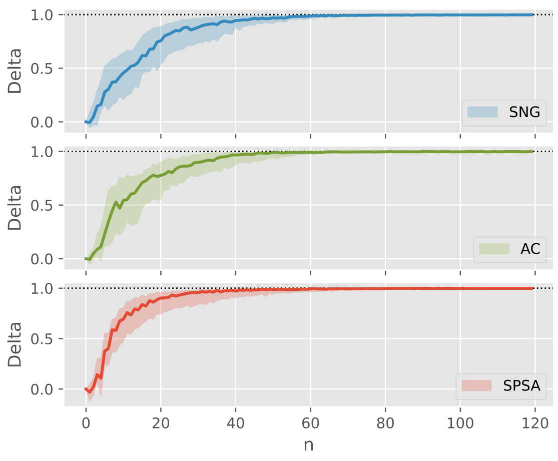
Figure B.3: Case 1: Median comparisons with no noise and high noise. Percentiles are shown in Figure B.4.



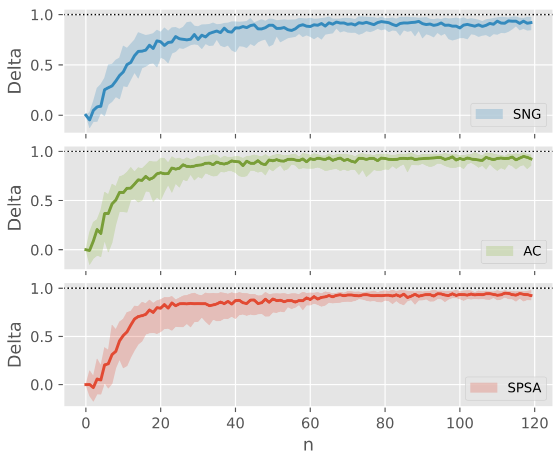
(a) 1D: No noise



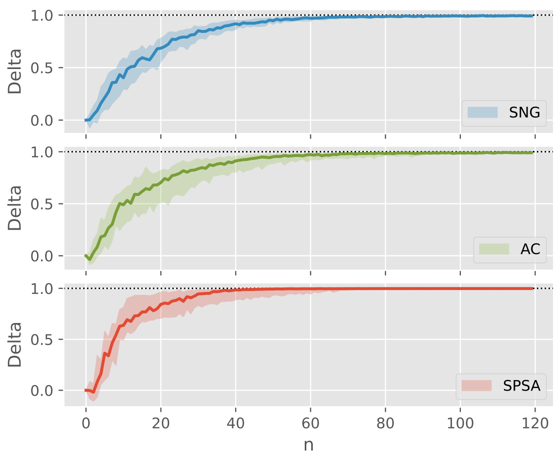
(b) 1D: High noise



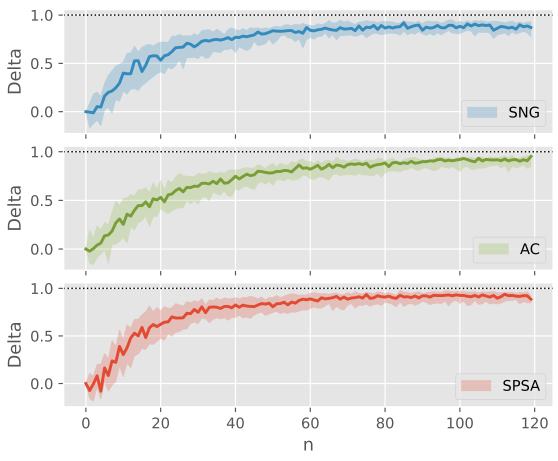
(c) 3D: No noise



(d) 3D: High noise

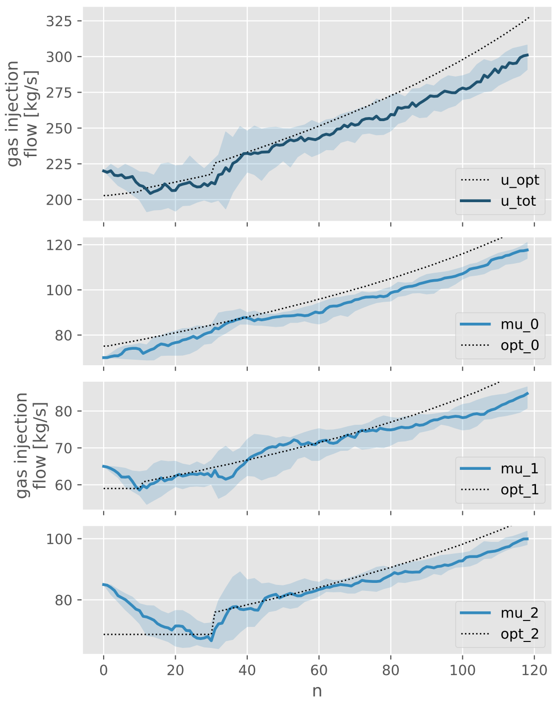


(e) 5D: No noise

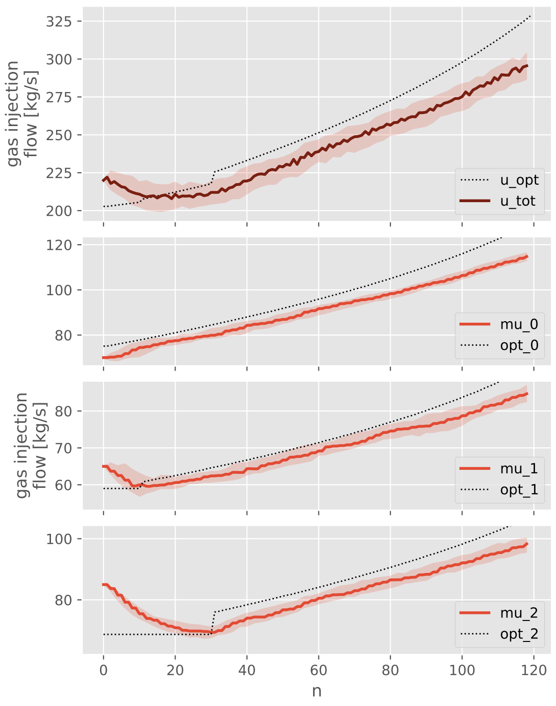


(f) 5D: High noise

Figure B.4: Case 1: Medians and percentiles with no and high noise (ref. Figure B.3 and Tables 5.4 and 5.5)

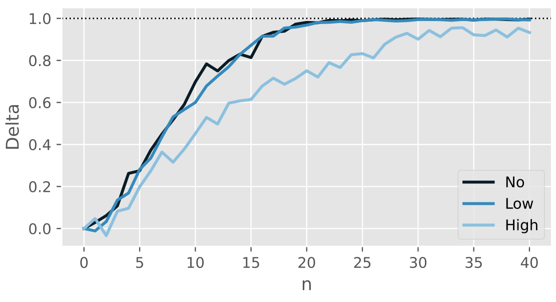


(a) 3D: SNG, new configuration

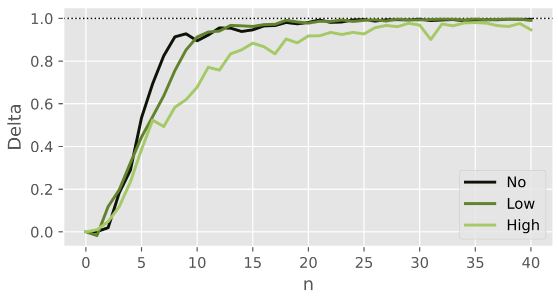


(b) 3D: SPSA, new configuration

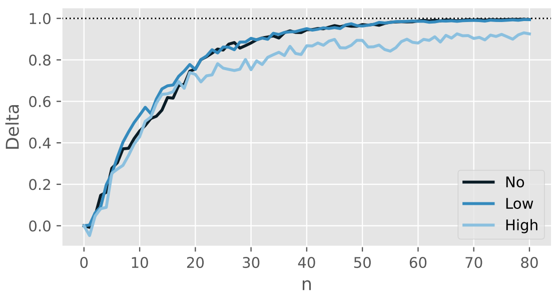
Figure B.5: Case 2: SNG and SPSA inputs (u) and means (mu) tracking the optimum with $M = 3$. The optimal inputs are marked in dotted lines.



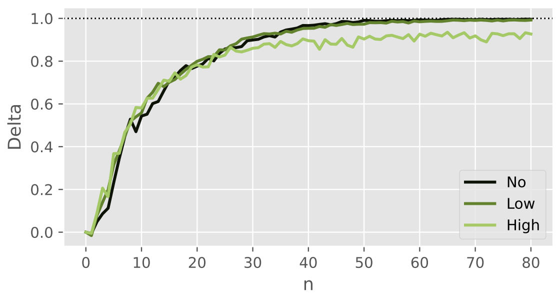
(a) 1D: SNG



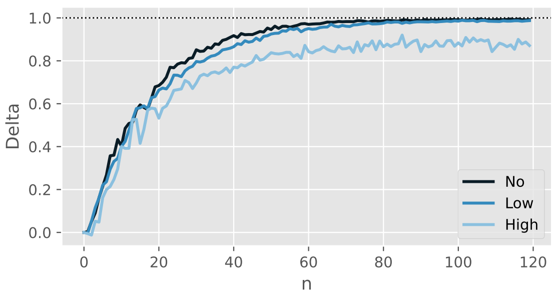
(b) 1D: AC



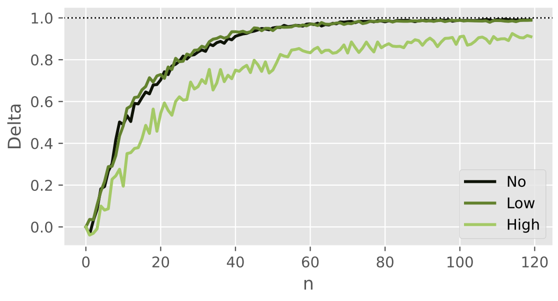
(c) 3D: SNG



(d) 3D: AC

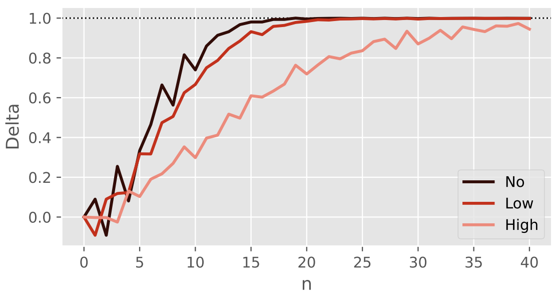


(e) 5D: SNG

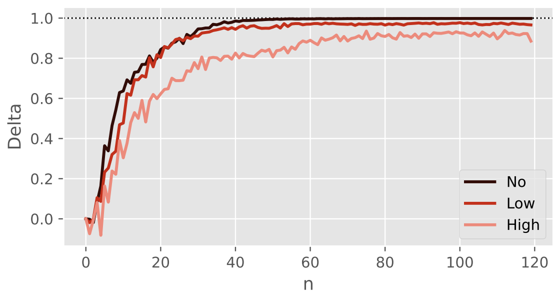


(f) 5D: AC

Figure B.6: Case 1: Comparisons of SNG and AC under the three different levels of noise. Lighter colors indicate more noise.

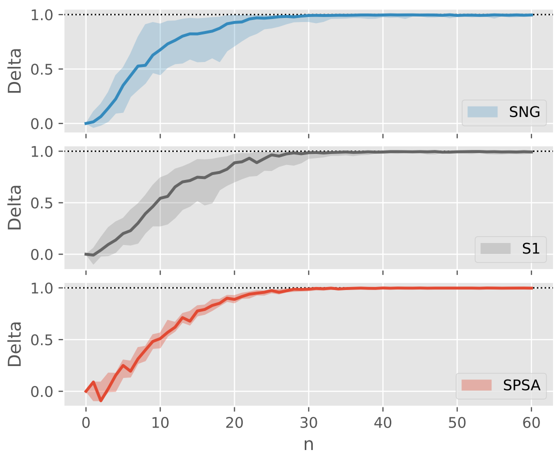


(a) 1D: SPSA

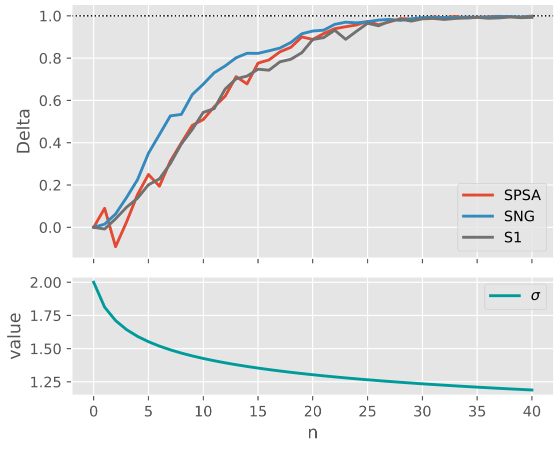


(b) 5D: SPSA

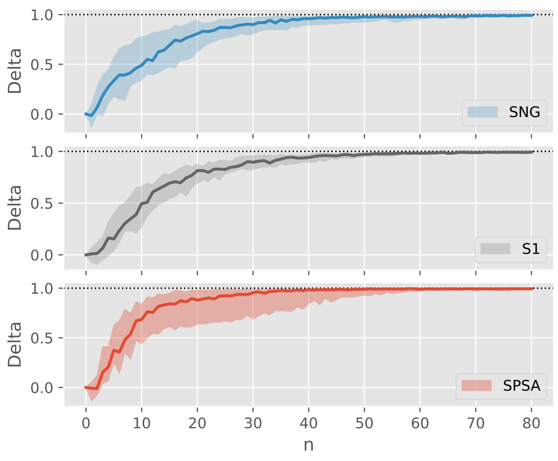
Figure B.7: Case 1: SPSA under different levels of noise. Lighter colors indicate more noise.



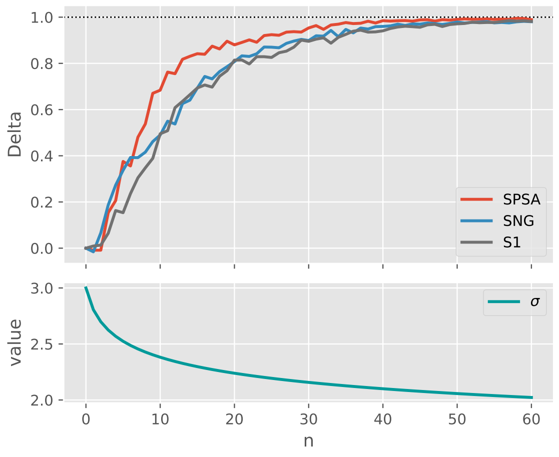
(a) 1D: Medians and 80-20 percentiles



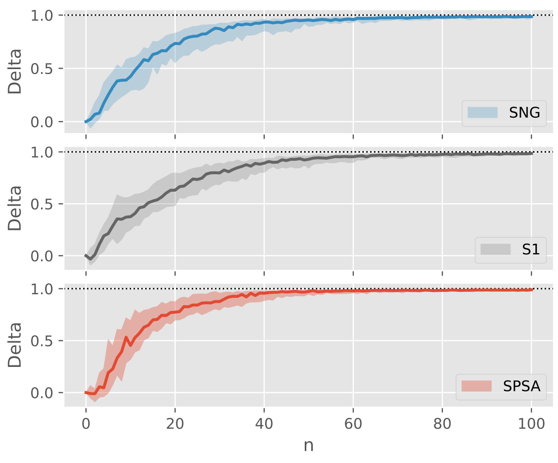
(b) 1D: Medians and σ_n, b_n with $\beta = 0.15$



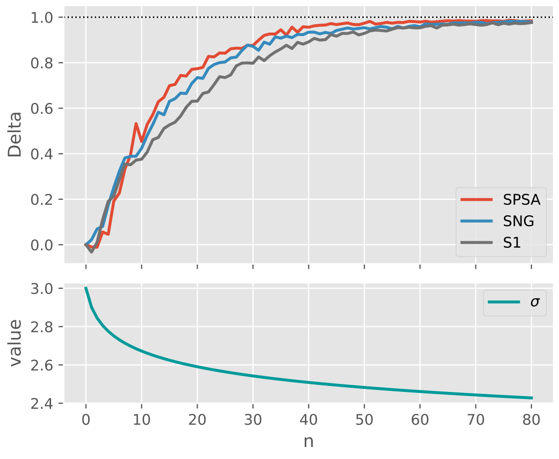
(c) 3D: Medians and 80-20 percentiles



(d) 3D: Medians and σ_n, b_n with $\beta = 0.10$



(e) 5D: Medians and 80-20 percentiles



(f) 5D: Medians and σ_n, b_n with $\beta = 0.05$

Figure B.8: Case 1: Fixed-variance optimization for S1, SNG and SPSA. Low noise. Figures (b) and (f) are duplicates of Figure 5.8.

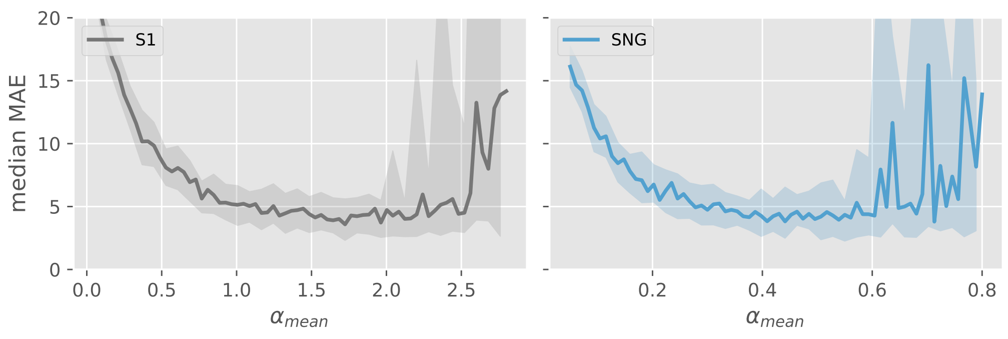
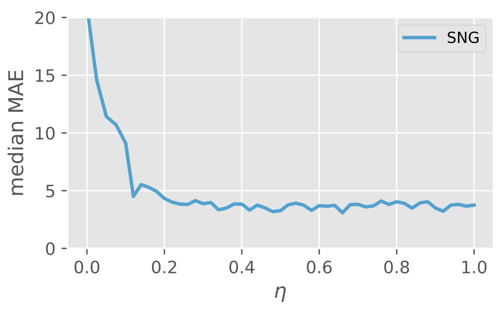
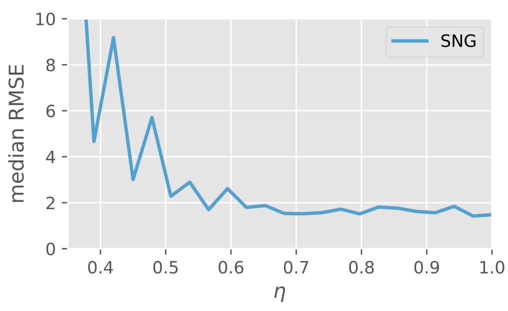


Figure B.9: Case 1: Sensitivity of α_σ for S1 and SNG. Comparing median MAE to parameter value. Logarithmic scale on x-axis.

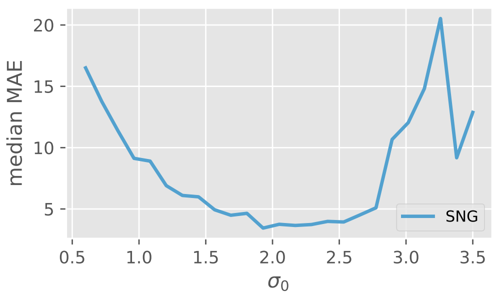


(a) 3D: Static optimum

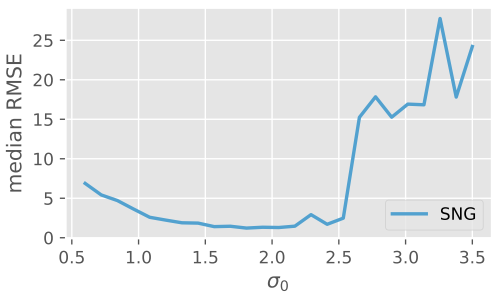


(b) 3D: Dynamic optimum

Figure B.10: Case 1 and 2: Sensitivity of average-reward weight η for SNG. Median over 30 simulations.

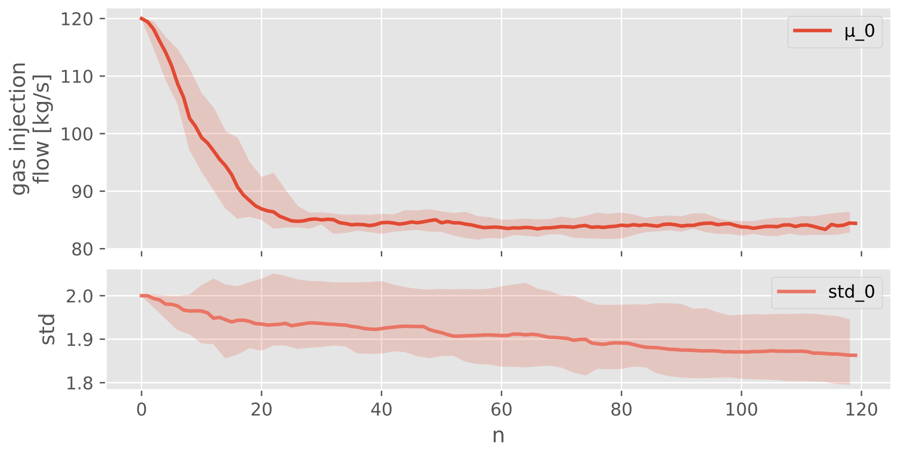


(a) 3D: Static optimum

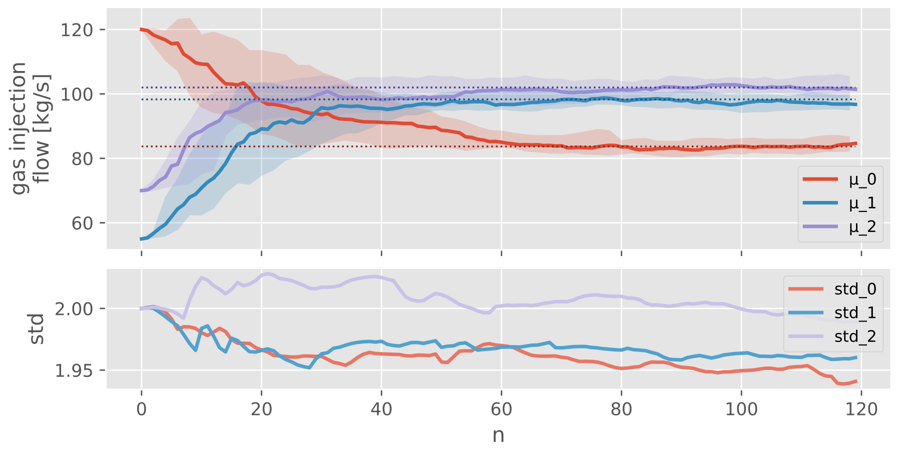


(b) 3D: Dynamic optimum

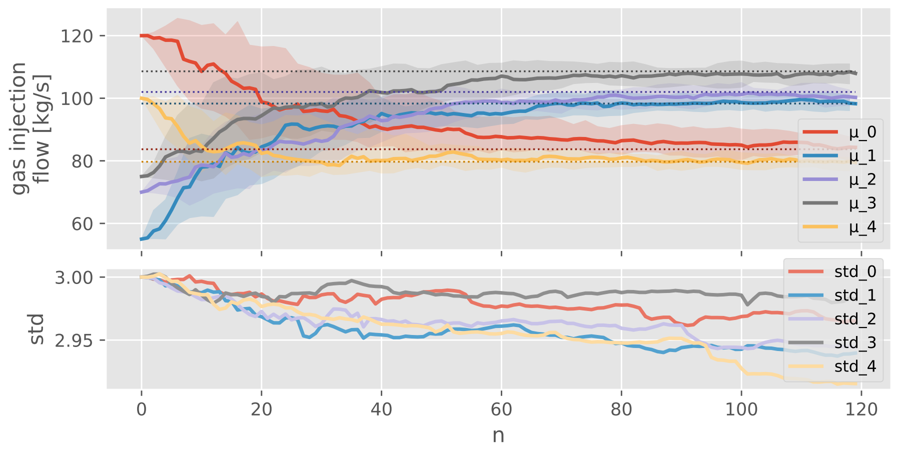
Figure B.11: Case 1 and 2: Sensitivity of initial SD $\boldsymbol{\sigma}_0$ for SNG. Median over 30 simulations.



(a) 1D

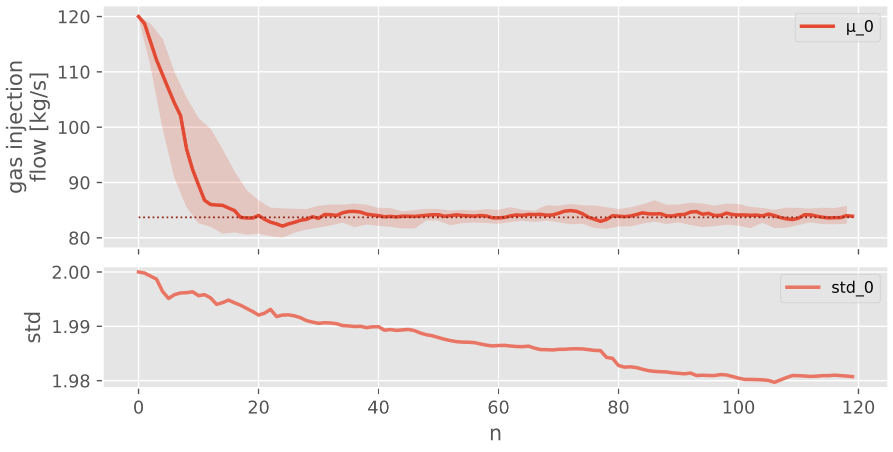


(b) 3D

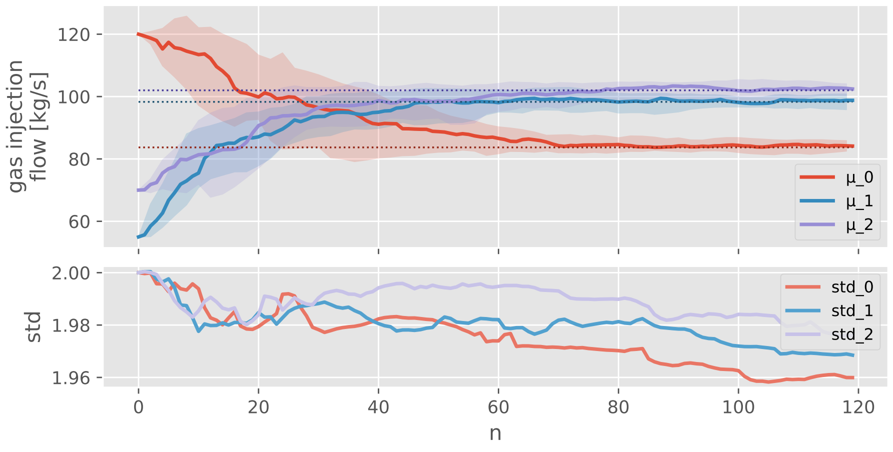


(c) 5D

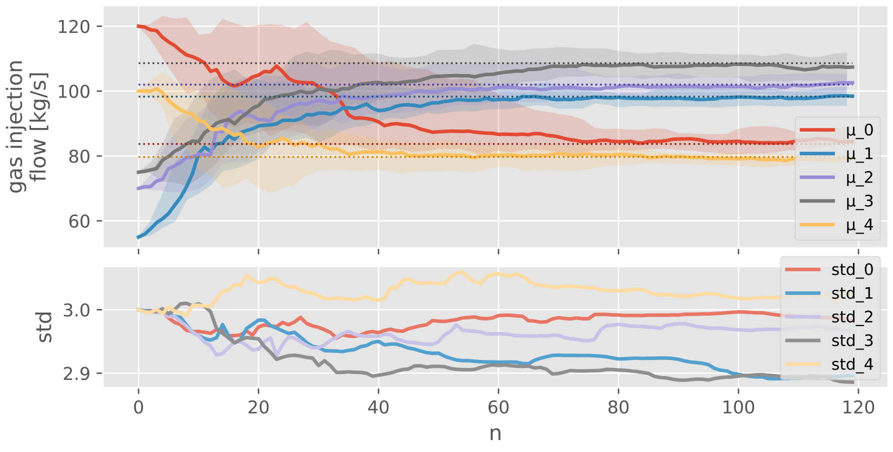
Figure B.12: Case 1: SNG policy means and SDs with low noise (from Table 5.3)



(a) 1D

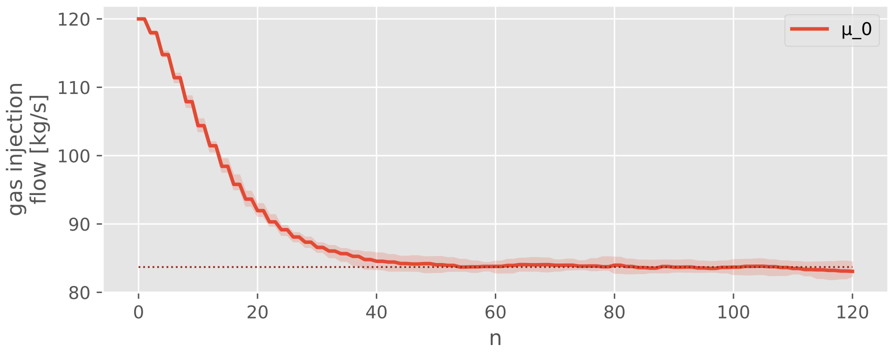


(b) 3D

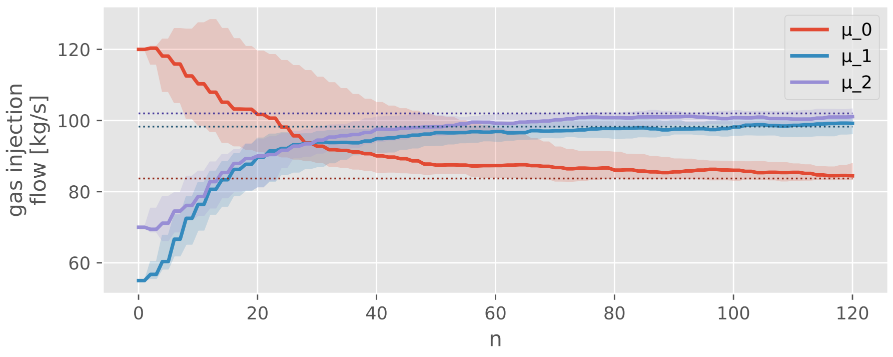


(c) 5D

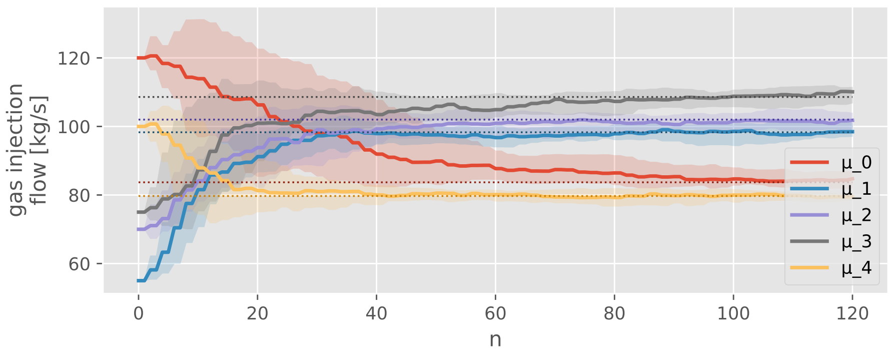
Figure B.13: Case 1: AC policy means and SDs with low noise (from Table 5.3)



(a) 1D



(b) 3D



(c) 5D

Figure B.14: Case 1: SPSA policy means with low noise (from Table 5.3)