Model-Free Safe Exploration and Optimization

Master's Thesis

Yang Wang

Department of Electrical and Computer Engineering
Chair of Control Systems

Supervisors:
Prof. Dr.-Ing. Daniel Görges
Prof. Dr.-Ing. Giancarlo Ferrari Trecate
Doctoral Assistant Baiwei Guo

March 31, 2022

Affidavit (Eidesstattliche Erklärung)

This thesis has been prepared on initiative and under guidance of my advisors. For the preparation I have not used others than the indicated aids.

Kaiserslautern, March 31, 2022

Yang Wang

Abstract

For many optimization problems in applications ranging from science, engineering, to economics, etc., objectives and constraints may not be explicitly known by mathematical expressions, but can only be accessed as outputs of black-boxes, such as simulations or experiments. The fundamental ingredient in traditional optimization methods, the derivative information, is thus not available. Such a setting encourages the use of model-free optimization methods, which do not require the availability of system models. Instead, these methods determine the next point to evaluate using previous input and output information from the systems. The choice of the next evaluation point is the essence of model-free optimization methods. In addition to the model unavailability issue, in many real-world applications, safety and the ability to operate under various constraints are critical prerequisites. The main concern of safety lies in the issue that the operation of systems in constraint-violated conditions may either not be able to perform or cause severe consequences such as device damages or human injuries. The safety requirement is typically crucial in applications such as medical treatments, power system operations, or robotic platforms. In model-free optimization scenarios, the safety constraints require the optimizer never evaluate decision points outside of certain safety areas. Recently, some safety-aware model-free optimization methods are proposed, such as the safe Bayesian optimization (SafeOpt-MC) [BKS21] and the safe logarithmic barrier method (0-LBM) [UKK19]. Though the SafeOpt-MC method enjoys evaluation-efficiency, it suffers from a high computational cost and limited scalability to high dimensional problems. The 0-LBM method on the other hand hardly approaches an optimal point when it is close to a boundary due to exponentially decreasing step sizes. In this thesis, we propose a novel model-free safe optimization algorithm, the safe-line-search optimization (0-SLS). This method extends the conventional line-search method to account for model-free, safety-critical circumstances. Given an initial safety-feasible point, the algorithm minimizes the objective function and evaluates only points that satisfy all safety constraints. To this end, with a low computational cost the algorithm carefully searches along a descent direction utilizing smoothness properties of underlying objective and constraint functions. The algorithm is empirically tested on several numerical examples and presents general outperformance than existing safe optimization methods. Finally, the algorithm is applied to a large-scale AC Optimal Power Flow optimization problem, and it demonstrates a comparable performance to the benchmark results. The code of the algorithm is available[1].

Keywords: Model-free optimization, Derivative-free optimization, Safe optimization, Line-search method, Power system, Optimal power flow problem.

[1] An example code is available at: `https://github.com/yangwangg/safe_line_search_optimization`

Kurzfassung

Bei vielen Optimierungsproblemen aus den Bereichen Wissenschaft, Technik, Wirtschaft usw. sind die Ausdrücke von Ziel- und Nebenbedingungsfunktionen manchmal nicht explizit bekannt, sondern nur als Ausgaben einer Blackbox wie einer Simulation oder zugänglich ein Experiment. Der grundlegende Bestandteil herkömmlicher Optimierungsmethoden, die abgeleiteten Informationen, ist in diesem Fall nicht verfügbar. Eine solche Einstellung fördert den Einsatz modellfreier Optimierungsmethoden. Diese Methoden erfordern nicht die Verfügbarkeit der Modelle für die Optimierungsprobleme. In vielen realen Anwendungen sind Sicherheit und die Fähigkeit zum Betrieb bei Vorhandensein von Einschränkungen, die sich aus einer Umgebung ergeben, kritische Voraussetzungen, da ein Betrieb von Systemen außerhalb einiger Sicherheitsbeschränkungen möglicherweise nicht durchgeführt werden kann oder schwerwiegende Folgen haben kann, wie z Geräteschäden oder Personenschäden. Diese Überlegung ist typischerweise in Anwendungen wie medizinischen Behandlungen, Betrieb von Stromversorgungssystemen oder Roboterplattformen erforderlich. Diese Einschränkungen erfordern, dass die Optimierungsverfahren sicherheitsgewährleistet sind, dh Entscheidungspunkte außerhalb bestimmter sicherer Bereiche werden niemals bewertet. Kürzlich wurden einige sicherheitsbewusste modellfreie Optimierungsverfahren vorgeschlagen, wie das SafeOpt-MC-Verfahren, das das Bayes'sche Optimierungsverfahren für mehrere eingeschränkte, sicherheitskritische Optimierungsprobleme verwendet, und das 0-LBM-Verfahren, das eine Log-Barrieren-Funktion verwendet Sicherheit zu garantieren. Obwohl das SafeOpt-MC-Verfahren evaluierungseffizient ist, leidet es jedoch unter einem großen Rechenaufwand und einer begrenzten Skalierbarkeit für hochdimensionale Probleme, und das 0-LBM-Verfahren nähert sich kaum einem optimalen Punkt mit exponentiell abnehmender Schrittgröße, wenn es nahe daran ist eine Grenze. In dieser Arbeit wird ein neuartiger modellfreier sicherer Optimierungsalgorithmus entwickelt, der die herkömmliche Liniensuchmethode erweitert, um einen modellfreien, sicherheitskritischen Umstand zu berücksichtigen. Ausgehend von einem anfänglichen sicherheitsrelevanten Punkt minimiert der Algorithmus eine objektive Funktion unter mehreren Sicherheitsbeschränkungen ohne explizite Kenntnis der Funktionen und wertet nur Punkte aus, die alle Sicherheitsbeschränkungen erfüllen. Zu diesem Zweck sucht der Algorithmus mit geringem Rechenaufwand sorgfältig entlang einer Abstiegsrichtung der Zielfunktion. Der Algorithmus wird an zwei Studienfällen empirisch getestet und mit bestehenden sicheren Optimierungsmethoden verglichen. Der Algorithmus wird auch angewendet, um ein groß angelegtes Optimierungsproblem des optimalen AC-Leistungsflusses bei der Untersuchung der Energiesystemanalyse zu lösen, und er zeigt eine vergleichbare Leistung mit dem Benchmark-Ergebnis.

Keywords: Modellfreie Optimierung, Ableitungsfreie Optimierung, Sichere Optimierung, Liniensuchverfahren, Optimales Kraftflussproblem.

Contents

List of Figures

List of Tables

List of Algorithms

Notation, Indices and Abbreviations

Notation

$\boldsymbol{x}$	Decision variable vector
d	Dimension of the decision variable vector
m	Number of constraints
$f^0(\boldsymbol{x})$	Objective function
$f^i(\boldsymbol{x})$	Constraint functions
$\|\cdot\|_1$	l_1-norm on $\mathbb{R}^d$
$\|\cdot\|_2$	l_2-norm on $\mathbb{R}^d$
L	Lipschitz constant
M	Smoothness constant

Indices

$\boldsymbol{x}_k$	$\boldsymbol{x}$ at discrete time k
$\boldsymbol{e}_j$	j-th unite coordinate vector
$f^i(\boldsymbol{x})$	i-th constraint function
$f^0(\boldsymbol{x})$	Objective function

Abbrevations

AC-OPF	AC optimal power flow
DC-OPF	DC optimal power flow
DFO-TR	Derivative-free optimization using trust region
e-SLS	Safe Line Search optimization with exact measurements
n-SLS	Safe Line Search optimization with noisy measurements
SafeOpt-MC	Safe Bayesian optimization algorithm for multiple constraints
s0-LBM	Safe Logarithmic Barrier method with Stochastic Zero-th Order Oracle
0-LBM	Safe Logarithmic Barrier method with Exact Zero-th Order Oracle

1 Introduction

1.1 Model-free safe optimization problem

Optimization problems appear widely in science, engineering, economics areas. Solving an optimization problem aims to find a set of *decision variables* that minimize or maximize an *objective function* under some *constraints*. The set of applicable decision points for an optimization problem is called the feasible region of this problem.

Conventionally, optimization problems are solved based on the models of the applications, which are analytical expressions of the objective and constraint functions. However, with increasingly complex and diverse problems, the objective and/or constraints sometimes cannot be explicitly expressed by mathematical equations, while only the function values as an output of a black-box can be accessed. The optimization problem that does not require system models is called the model-free or black-box optimization problem. Existing strategies for solving this problem can be generally categorized into three classes: heuristic methods, derivative-free methods, and Bayesian optimization.

The derivative-free methods are those methods that do not require the derivative information about the objective and constraint functions. Nowadays, derivative free methods are frequently used in many applications, for example by Google for auto-tuning of parameters of machine learning models, or for controller configuration of drones in an unknown environment. The idea of derivative-free optimization is to iteratively query function evaluations to a black-box such as an unknown simulation or experiment, and then utilize the received function values to infer the next decision variables until a convergence condition is satisfied.

Another important concept in optimization problems is the *safety*. Safety in an optimization process means that all function evaluations during optimization shall not violate constraints, as it may cause significant losses such as equipment damages or personal injuries. Safety is especially important in black-box optimization, as underlying boundaries are unknown to the optimizer, and during pursuing the optimal solution, it needs to carefully choose the decision point to ensure that no unsafe evaluations are made. This setting is also called the *safe learning* in the machine learning community. For such a problem, the safe threshold can be treated as a *safety constraint*, and one shall guarantee the safety constraints are not violated during the optimization process.

1.2 Literature review

1.2.1 Derivative-free optimization problem

The derivative-free optimization, also known as the black-box optimization, the zero-th order optimization, is a comprehensive topic. There exist many strategies for solving this problem in the literature. For solving deterministic optimization problems, there are deterministic methods and randomized methods. The deterministic methods can be classified into two classes: direct-search method and model-based method. The direct-search method measures a set of function values and determines the next decision point by comparison of these values, whereas, the model-based method uses an estimated model of underlying functions to find the candidate points. With an estimated model, the next decision point is mainly computed by a model-based trust-region method or a line search method. The randomized optimization methods randomly evaluate underlying functions at points in the feasible region and accept only the points reducing the objective function.

1.2.2 Direct search method

The term 'direct search' is originated from [HJ61]. Followed the convention of [Wri96], a direct-search method is the method that only uses function values and does not approximate its gradient. One of the direct-method is the simplex method. This method manipulates a set of $n + 1$ affinely independent points, such as, the vertices of a simplex, when solving deterministic optimization problem. An extension of the simplex method is the Nelda-Mead method. [NM65] extends the operations of "expansion" and "contraction" in addition to the "reflection" and "shrink" operations proposed by [SHH62]. The new operations enable the Nelder-Mead simplex method to distort the simplex so that it can account for possible curvature of the objective function. The Nelder-Mead method has an incredible popularity mostly because that it has been included in *Numerical Recipes* [Pre+96]. The Nelder-Mead method is also the algorithm underlying the *fminsearch* function in MATLAB. The performance of Nelder-Mead mehtod in practice is benchmarked in [RS13].

Another direct-search method is the directional direct-search (DDS) method. This method evaluates function values at a finite set of points around the current point x_k. These points are selected by adding terms $\alpha_k \boldsymbol{d}$ to the current point, where α is a positive step length and $\boldsymbol{d}$ is an element of a finite set of directions $\boldsymbol{D}_k$. The next iterate x_{k+1} is selected to be the point that produces a (sufficient) decrease in the objective function. If such a point exists, the step size is possibly increased. If there is no point produces sufficient decrease, the next point is x_{k+1}

is set to x_k and the step length is decreased. The convergence of DDS method is firstly proved in [Jea71].

1.2.3 Model-based trust-region method

Another class of deterministic derivative-free optimization method is model-based method. In the context of derivative-free optimization, the model-based method determines the next iterate $\boldsymbol{x}_{k+1}$ using an approximate model m_k of the objective function f. This method usually assumes some smoothness in the objective function so that a smooth model can be constructed. The most commonly used approximation models in derivative-free optimization are polynomial models. To construct an approximate polynomial model, a set of $(n+2)(n+1)/2$ points that are dispersed enough in the region of interest are evaluated $\boldsymbol{Y}_k = \left[f(\boldsymbol{x}_k^1), \ldots, f\left(\boldsymbol{x}_k^{(n+2)(n+1)/2}\right)\right]^{\mathrm{T}}$. The polynomial model is then constructed by a linear combination of elements in a polynomial space $\boldsymbol{\Phi}(\boldsymbol{x}) = \left[\phi_1(\boldsymbol{x}), \ldots, \phi_{(n+2)/(n+1)/2}(\boldsymbol{x})\right]^{\mathrm{T}}$ with a coefficient vector $\boldsymbol{a}_k$ by $m_k = \boldsymbol{\Phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{a}_k$. By solving the equation $\boldsymbol{Y}_k = \boldsymbol{\Phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{a}_k$, the coefficient vector $\boldsymbol{\alpha}_k$ can be determined. The quadratic interpolation model has been used for more than fifty years [Win70] and are employed with a series of model-based methods, for example, methods in [CT96], [CST97] and [Pow98].

With the constructed approximation model, a general framework of model-based method is the derivative-free trust-region (DFO-TR) method. Traditional trust region method defines a region around the current iterate $\boldsymbol{x}_k$ in which a quadratic model for the objective function is constructed. The candidate of the next iterate $\boldsymbol{x}_{k+1}$ is chosen as an approximate minimizer of the approximation model in the trust region. A ρ-test is made for the candidate point, in which they measure the ratio between the actual decrease in the objective function evaluated at the candidate point and the estimated value computed from the model. The candidate is accepted, if there is a well agreement between the actual reduction and the estimated one. If the candidate is not acceptable, the size of trust region is reduced, and a new model is constructed and a new candidate is found by minimizing the new model. In derivative-free case, by doing additional test, the trust region is avoided to shrink to zero solely because of poor representation of the objective function by the approximate model. With the augmented test, the trust region radius going to zero indicates the convergence of the objective function. Under reasonable smoothness assumption, the DFO-TR method is shown to have first-order convergence with a well-known proof given by [CSV09]. A pseudocode of DFO-TR method is shown in Algorithm 1.

Algorithm 1 Derivative-free optimization with trust-region (Pseudocode)

Input: Initial trust region radius Δ_0, polynomial space $\boldsymbol{\Phi}(\boldsymbol{x})$;
for $k = 1, 2, \ldots,$ **do**
    Evaluate $\frac{1}{2}(n+2)(n+1)$ points $\boldsymbol{Y}_k$ around the current iterate $\boldsymbol{x}_k$;
    Form a quadratic model m_k by interpolation $\boldsymbol{Y}_k = \boldsymbol{\Phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{a}_k$;
    Compute a candidate point $\hat{\boldsymbol{x}}_{k+1}$ by minimizing m_k;
    Compute $\rho = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_{k+1})}$, do a ρ-test;
    if ρ-test passes **then**
        Accept candidate $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}_{k+1}$;
        Update Δ_k
    else
        Reduce Δ_k, Update $\boldsymbol{Y}_k$;
        Go to Step 4;
    end if
    Evaluate $f(\boldsymbol{x}_{k+1})$;
    if Convergence condition is satisfied **then**
        Algorithm Stop
    end if
end for

1.2.4 Model-based line-search method

While the majority methods in derivative-free optimization can be classified as direct-search methods or model-based trust-region methods, some works try to hybridize these two classes, for example, [DT97], [CDV08] and [CL13].

Several derivative-free optimization methods are developed based on line-search (LS) methods. In line-search methods, they first determine a search direction that possibly minimizes the objective, then efforts are focused on finding a suitable step length along the search direction. The line-search methods usually provide a monotonous decrease in the objective function. [GLL88] and [DGG84] propose condition for a suitable step length, and they provide methods for constructing such steps. [LS02] proposes a convergent method that combines line-search with pattern-search method.

In derivative-free optimization, some considerations are taken in non monotone line-search methods. As the gradient is not available, the search direction at each iteration may not necessarily be a descent direction. A non-descent step can be employed to provide a global convergence result. [GS07] extends the traditional line search method by coordinate search and [BB88] develops a derivative-free two-point step length method that has non-monotone global convergence property. [GR15] proposes an extension of such non-monotone method to a wider range of algorithms that employ simplex gradients. This work presents

an union of the direct-search and model-based derivative-free optimization methods.

[UKK19] develops a novel line search method (0-LBM) for derivative-free constrained optimization problem. They construct a log-barrier function $B_\eta(\boldsymbol{x}) = f^0(\boldsymbol{x}) - \eta \sum_{i=1}^{m} \log(-f^i(\boldsymbol{x}))$ that combining the objective and constraint functions together. The optimization problem is solved by estimating the gradients of the log-barrier function with finite-difference method and progress along the descent direction with a selected safe step length computed using an local smoothness constant L_2. With assumptions on the smoothness of the unknown functions $f^i(\boldsymbol{x})$, they guarantee that the algorithm are safe, in terms of that the constraints are not violated during the optimization process and that it finds a η-approximate KKT point with a upper bound on the number of required iterations. A pseudocode of the 0-LBM method is shown in Algorithm 2.

Algorithm 2 Safe Logarithmic Barrier method (Pseudocode)

Input: Initial feasible point $\boldsymbol{x}_0$, smoothness constant L, $M > 0$;
for $k = 1, 2, \ldots,$ **do**
    Compute a safe difference length ν_k;
    Estimate gradients of unknown functions $f^i(\boldsymbol{x})$ by finite-difference with length ν_k;
    Compute a gradient estimator $\boldsymbol{g}_k$ for $\nabla B_\eta(\boldsymbol{x}_k)$ and a local smoothness constant L_2 for $B_\eta(\boldsymbol{x}_k)$;
    Select a safe step length γ_k;
    Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k \boldsymbol{g}_k$;
    if Convergence condition is satisfied **then**
        Algorithm Stop
    else
        Evaluate $f^i(\boldsymbol{x}_{k+1})$
    end if
end for

1.2.5 Randomized optimization method

In addition to deterministic methods, the randomized optimization method has also promising properties. There are two popular randomized methods: pure random search and Nesterov random search.

Pure random search is a nature method in randomized derivative-free optimization. In pure random search method, a point within a predefined region is generated by a random point generation scheme. The algorithm is updated with the generated point if it produces a descent in the objective function, otherwise, the generation scheme is called for a new point. If the random point generation

is independent on the function evaluation, the entire set of points used in pure random search method can be generated beforehand. The convergence of pure random search can be proved by the fact that after N iterations, the probability that there exists a point $\hat{x}_k$ arbitrary near to the global optimum converges to 1, provided that the random point generation scheme ensures that the probability in N iterations no point is near to the global optimal is zero. The pure random search method is popular for several reasons: firstly, it is easy to implement, secondly, it exhibits good scaling property by possibility of evaluating many points simultaneously. thirdly, it require only few assumption about the objective function. [ZS92] and [PSZ89] made heuristic modifications on pure random search method for certain types of problems to improve the empirical performance while maintaining its global optimization property.

The Nesterov random search is proposed by [NS17]. This method is largely motivated by Gaussian smoothing. With a Gaussian smooth function encoding the objective function, the gradient of f can be approximated by computation of expectation over Gaussian random vector $\boldsymbol{u} \in \mathbb{R}^n$ weighted by finite difference $f(\boldsymbol{x} + \mu\boldsymbol{u}) - f(\boldsymbol{x})$ and inversely weighted by Gaussian radial distance from $\boldsymbol{x}$. Nesterov and Spokoiny propose a collection of random gradient-free oracles $\boldsymbol{g}_\mu(\boldsymbol{x}; \boldsymbol{u})$ with smoothing parameter $\mu \geq 0$. Then the next iterate is chosen by $x_{k+1} = \text{proj}(\boldsymbol{x}_k - \alpha_k \boldsymbol{B}^{-1}\boldsymbol{g}(\boldsymbol{x}_k; \boldsymbol{u}_k), \boldsymbol{\Omega})$, where $\text{proj}(\cdot, \boldsymbol{\Omega})$ is the projection on feasible region $\boldsymbol{\Omega}$.

Randomization is also implemented in directional direct search framework. In the randomized DDS method, the candidate directions are randomly sampled from some distribution at each iteration. This work aims to reduce the $\mathcal{O}(n)$ function evaluations per-iteration to $\mathcal{O}(1)$, while maintaining some form of global convergence of original DDS method. [Bib+20] propose a randomized DDS method, in which the candidate search direction at each iteration is $\boldsymbol{D}_k = \{\boldsymbol{e}_i, -\boldsymbol{e}_i\}$, where $\boldsymbol{e}_i$ is the i-th coordinate vector and is chosen from $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$ with probability propotional to the Lipschitz constant of i-th partial derivative of f.

[DMR08] propose a randomized DDS method that employs a non-monotone line search strategy on randomly selected search directions to tackle with the situation where a descent in the objective along the search direction may not initially apparent.

1.2.6 Stochastic optimization

Stochastic derivative-free optimization problem is the problem that aims to minimize an objective function with only access to stochastic measurements of function values. Generally, a stochastic optimization is defined as

Problem 1. ***Stochastic Optimization Problem***

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \quad f(x) = \mathbb{E}_\xi\left[\tilde{f}(x;\xi)\right] \tag{1.1}$$

$$\text{subject to} \quad x \in \Omega \tag{1.2}$$

where $x \in \mathbb{R}^d$, $\xi \in \Xi$ is an independent and identically distributed (i.i.d.) random variable, the function realization $\tilde{f}$ depends both on x and ξ, Ω is the feasible set for x.

The random variable ξ can be interpreted as an environmental variable that is out of control of optimization methods. This set-up occurs, for example, when the function measurements are corrupted by some environmental noises. The goal of stochastic optimization is to minimize the expectation of function realizations $\tilde{f}$ over random variable $\xi \in \Xi$.

In order to solve stochastic optimization problem, people have modified methods for deterministic objective to tackle the stochasticity. For the class of direct-search method, [Cha12] proposes modification on Nelder-Mead method that takes an increasing number of samples for candidate point and all other points in the simplex. [BI96] proposes a Nelder-Mead variant that avoids early-termination due to repeated shrinking. They suggest to reduce the degree that the simplex is shrunk, re-evaluate the best point after each shrinking, and re-evaluate all reflected points before doing a contraction. [KZ10] considers a variant of directional direct-search method that uses the sample mean of function evaluations with a increasing sample times. They present a consistent result and convergence properties of the DDS methods.

Researchers also modify the deterministic model-based methods for stochastic optimization problem. [DF06] employ a variant of derivative-free trust-region methods for stochastic optimization. They use Bayesian methods to dynamically update the budget of sample size of function evaluations. This budget is assigned to the current set of interpolation points so that the variance of function sample mean at each interpolation point is reduced.

[CHW13] propose a method that combines the response surface methodology [BD87] with trust-region. They suggest that with an increasing number of sample size, it is almost sure (with probability of 1) that the approximated model gradients equal the true gradient as iterate goes to infinity.

[LB16] and [CMS18] suggest to use a probabilistic fully linear model which says that the condition for fully linear model can be hold only with some probability [BSV14]. [AM17] propose a model-based trust-region method for constrained stochastic optimization problem. They deal with stochasticity using Gaussian-process-based models of robustness measures, for example, expectation

and conditional value at risk.

1.2.7 Bayesian Optimization

Bayesian optimization [MTZ78] is a black-box global optimization algorithm that does not simply rely on local gradient and Hessian approximation. Bayesian optimization assumes that the unknown function is sampled from a Gaussian process (GP) and the GP model is constantly updated after each function evaluation.

Two ingredients in the Bayesian optimization are the GP prior model and the $acquisitionfunction$. The GP prior model expresses the assumption about properties of the objective function, such as smoothness or probability distribution of function values in stochastic optimization. The $acquisitionfunction$ is an utility function, which measures the possible function reduction or function exploration in the next iteration. The decision point is chosen by optimizing the acquisition function. There are two commonly used acquisition function: the expected improvement (EI) over the best result measured until the current iteration, and the Gaussian process upper confidence bound (UCB).

The nature of probabilistic model for the unknown function has some appealing advantages. Firstly, it solves the stochastic optimization naturally with the probabilistic model on the unknown function. Secondly, it is able to find the optimal point of a non-linear, non-convex objective function with relatively few function evaluations. The cost for less function measurements is that it requires more computation to update the GP model and to compute the next decision point. The Bayesian optimization is typically suitable for problem when function evaluation is expensive. Algorithm 3 presents a pseudocode for Bayesian optimization

Algorithm 3 Bayesian optimization (Pseudocode)

1: **for** $k = 1, 2, \ldots,$ **do**
2: Compute x_{k+1} by optimizing an acquisition function α
$$x = \arg\max_x \alpha(x; \mathcal{D}_k);$$
3: Evaluate objective function for y_{k+1};
4: Augment data $\mathcal{D}_{k+1} = \{\mathcal{D}_k, (x_{k+1}, y_{k+1})\}$;
5: Update GP model;
6: **end for**

1.3 Thesis outline

This thesis is organized by the following: Chapter 2 formulates the proposed model-free safe optimization algorithm and compares the optimization performance with other algorithms by two simulation cases. Chapter 3 introduces the Optimal Power Flow problem and analyses the optimization result solved by the proposed algorithm. Chapter 4 concludes the thesis.

2 Safe-line-search optimization

2.1 Problem formulation

The model-free safe optimization problem is the problem in which both objective and constraint functions are unknown, and the constraints are safety-critical, which implies that no evaluation violating safety constraints throughout the optimization process shall be made. The model-free safe optimization can be formulated as:

Problem 2. ***Model-Free Safe Optimization Problem***

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}} \quad f^0(\boldsymbol{x}) \tag{2.1}$$

$$\text{subject to} \quad f^i(\boldsymbol{x}) \leq 0,\ i = 1, \ldots, m \tag{2.2}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the decision variable, the objective function $f^0(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ and the constraints $f^i(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ are not explicitly known and are possibly non-linear, non-convex functions. d is the number of dimensions of decision variable and m is the number of constraints.

In problem 2, both objective and constraint functions can only be evaluated in a safety feasible set $\mathcal{D}$. The safety feasible set $\mathcal{D}$ is defined as:

$$\mathcal{D} := \{\boldsymbol{x} \in \mathbb{R}^d : f^i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m\} \tag{2.3}$$

2.2 Assumptions

For solving model-free optimization problem, some assumptions about the objective and constraint functions are assumed.

Assumption 1. *Let $L, M \geq 0$, the constraints $f^i(\boldsymbol{x})$ for $i = 1, \ldots, m$ are L-Lipschitz continuous on $\mathcal{D}$ and the objective and constraint functions $f^i(\boldsymbol{x})$ for $i = 0, \ldots, m$ are M-smooth on $\mathcal{D}$.*

Assumption 2. *The feasible set $\mathcal{D}$ is not empty, and a known safety feasible starting point $\boldsymbol{x}_0$ is given for which $f^i(\boldsymbol{x}_0) \leq 0$, for $i = 1, \ldots, m$.*

2.3 Preliminaries

In this thesis, the following notations and definitions are used:

- Denote by $\|\cdot\|_1$ the l_1-norm on $\mathbb{R}^d$;
- Denote by $\|\cdot\|_2$ the l_2-norm on $\mathbb{R}^d$;

Definition 1. *A function $f(\boldsymbol{x})$ is called L-Lipschitz continuous on $\mathbb{R}^d$ with $L \geq 0$ if for all $\boldsymbol{x}$, $\boldsymbol{y} \in \mathbb{R}^d$:*

$$|f(\boldsymbol{y}) - f(\boldsymbol{x})| \leq L\|\boldsymbol{y} - \boldsymbol{x}\|_2 \tag{2.4}$$

Definition 2. *A function $f(\boldsymbol{x})$ is called M-smooth on $\mathbb{R}^d$ with $M \geq 0$ if the gradient $\nabla f(\boldsymbol{x})$ is M-Lipschitz continuous on $\mathbb{R}^d$:*

$$\|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|_2 \leq M\|\boldsymbol{y} - \boldsymbol{x}\|_2 \tag{2.5}$$

Definition 3 (sub-Gaussian random variable)**.** *A random variable X is called sub-Gaussian (subG) with variance σ^2, if $\mathbb{E}(X) = 0$ and for any $s \in \mathbb{R}$ its moment generating function satisfies:*

$$\mathbb{E}\left[\exp(sX)\right] \leq \exp(\frac{\sigma^2 s^2}{2}) \tag{2.6}$$

In optimization problems, the mathematical tool used to analyze the minimizers of smooth functions and bounds conditions for constraints is Taylor's theorem. It is the central tool for studying the safety constraints in this thesis. Taylor's theorem states the following.

Theorem 1 (**Taylor's theorem**)**.** *Let $f^i(\boldsymbol{x})$ be a continuously differentiable, and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, there is:*

$$f^i(\boldsymbol{y}) = f^i(\boldsymbol{x}) + \nabla f^i\left(\boldsymbol{x} + t\left(\boldsymbol{y} - \boldsymbol{x}\right)\right)^{\mathrm{T}}\left(\boldsymbol{y} - \boldsymbol{x}\right) \tag{2.7}$$

where $t \in (0, 1)$. If $f^i(\boldsymbol{x})$ is twice continuously differentiable, then:

$$f^i(\boldsymbol{y}) = f^i(\boldsymbol{x}) + \nabla f^i\left(\boldsymbol{x}\right)^{\mathrm{T}}\left(\boldsymbol{y} - \boldsymbol{x}\right) + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x})^{\mathrm{T}}\nabla^2 f^i\left(\boldsymbol{x} + t\left(\boldsymbol{y} - \boldsymbol{x}\right)\right)\left(\boldsymbol{y} - \boldsymbol{x}\right) \tag{2.8}$$

for $t \in (0, 1)$.

From Theorem 1, if $t = 0$ and using Assumption 1 about the function $f^i(\boldsymbol{x})$ to replace $\nabla f^i(\boldsymbol{x})$ and $\nabla^2 f^i(\boldsymbol{x})$ by L and M respectively, then from (2.7) there is:

$$f^i(\boldsymbol{y}) \leq f^i(\boldsymbol{x}) + L\|\boldsymbol{y} - \boldsymbol{x}\|_2 \tag{2.9}$$

and from (2.8), there is:

$$f^i(\boldsymbol{y}) \leq f^i(\boldsymbol{x}) + \nabla f^i(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{x}) + \frac{1}{2}M\|\boldsymbol{y} - \boldsymbol{x}\|_2^2 \tag{2.10}$$

Equations (2.9) and (2.10) will be used frequently in this thesis.

To determine if or not a point $\boldsymbol{x}_*$ is an (local) optimum point for the optimization problem, an optimality condition is used.

In constrained optimization problems, the necessary condition for $\boldsymbol{x}_*$ to be a (local) optimum of the problem is the *Karush-Kuhn-Tucker (KKT) condition.* This condition is formulated based on the *Lagrangian function* which is defined below:

Definition 4. *The Lagrangian function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ for the constrained optimization problem 2 is defined as:*

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f^0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda^i f^i(\boldsymbol{x}) \tag{2.11}$$

where $\boldsymbol{\lambda} \in \mathbb{R}_+^m$ is called the Lagrangian multipliers.

Using the Lagrangian function (2.11), the constrained optimization problem 2 is converted into an unconstrained problem:

$$\underset{\boldsymbol{x},\ \boldsymbol{\lambda}}{\text{minimize}}\ \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \tag{2.12}$$

The optimality condition for constrained optimization problem 2 and the equivalent unconstrained optimization problem (2.12) is formulated as the KKT condition.

The KKT condition holds in the presence of a regularity condition of *Linear independence constraint qualification (LICQ).* It is defined as:

Definition 5. *The linear independence constraint qualification (LICQ) for the safe optimization problem 2 holds if constraint gradients $\nabla f^i(\boldsymbol{x})$, $i = 1, \ldots, m$ are linearly independent.*

With the definitions 4 and 5, the KKT condition is defined as:

Theorem 2 (**Karush-Kuhn-Tucker condition**). *Suppose that* $f^i(\boldsymbol{x})$, $i = 0, \ldots, m$ *are continuously differentiable, and that the LICQ holds for constraints* $f^i(\boldsymbol{x})$, $i = 1, \ldots, m$. *A point* $\boldsymbol{x}_*$ *is a local minimizer of the objective function* $f^0(\boldsymbol{x})$ *if there exists a set of Lagrangian multipliers* λ_*^i, $i = 1, \ldots, m$ *such that the following conditions hold:*

$$\|\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_*, \lambda_*^i)\|_2 = 0, \ \forall i = 1, \ldots, m \tag{2.13}$$

$$f^i(\boldsymbol{x}_*) \leq 0, \ \forall i = 1, \ldots, m \tag{2.14}$$

$$\lambda_*^i f^i(\boldsymbol{x}_*) = 0, \ \forall i = 1, \ldots, m \tag{2.15}$$

$$\lambda_*^i \geq 0, \ \forall i = 1, \ldots, m \tag{2.16}$$

In Theorem 2, (2.13) is the stationary equation, (2.14) defines the primary feasibility, (2.15) is the complementary slackness and (2.16) defines the dual feasibility.

The KKT condition is a first-order necessary optimality condition as it considers only the properties of the gradients of the objective and constraint functions. A point that satisfies the KKT condition is called a KKT point.

With the required preliminaries, a model-free safe optimization algorithm called the Safe-Line-Search (SLS) method is proposed. This algorithm considers two situations: first, the function measurements are exact (e-SLS); second, the function measurements are corrupted by noises (n-SLS). In the following sections, algorithms for each case are formulated.

2.4 Main idea of the algorithm

The model-free safe-line-search algorithm employs a line search optimization scheme using estimated gradients of unknown objective and constraint functions while satisfying safety constraints. In this algorithm, at each iteration, gradients of unknown functions are estimated using finite-difference method with a safety-guarantee difference step length, and a descent search direction is computed using the gradient estimators. Along the search direction, a step length is carefully chosen such that a sufficient decrease in the objective can be expected and all safety constraints will not be violated. The iteration is repeated until a convergence condition is satisfied. The safe-line-search algorithms can be summarized by the following steps:

1. Estimate function gradients;
2. Find a search direction;
3. Compute a safety-guarantee step length;
4. Update new decision point and check the convergence condition;

The algorithm is first formulated for the case of exact measurements, then is extended to the case of noisy measurements with modifications to deal with uncertainties involved in function measurements. Section 2.5 introduces the safe-line-search algorithm for exact measurements. Section 2.6 describes the modifications of algorithm for the case of noisy measurements. Section 2.7 discusses the convergence and safety properties of the algorithm.

2.5 Safe-line-search algorithm for exact measurements (e-SLS)

2.5.1 Section structure

The structure of this section follows the formulation step of the algorithm. The safe-line-search algorithm firstly estimates the gradients of the underlying functions, then a search direction is computed according to estimated gradients. A safe set is then defined so that the next point within this set is safety guarantee. If the optimization point is close to a boundary, the initial search direction is projected onto the perpendicular direction of gradients of the active constraints. Finally, a step length is selected along the computed search direction within the formulated safe set. The iteration is repeated until a convergence condition is met.

2.5.2 Gradient estimator

In model-free optimization problem, the objective and constraint functions are only accessible as function values returned from a black-box simulation or experiment. The important information for optimization, the function gradient, is not directly available. While, the gradients of unknown functions can be estimated using functions measurements by *finite difference method*. Concretely, at point $\boldsymbol{x}_k \in \mathbb{R}^d$, the gradient of a multivariate function $\nabla f^i(\boldsymbol{x}_k)$ can be estimated by:

$$G^i(\boldsymbol{x}_k, \nu_k) = \sum_{j=1}^{d} \frac{f^i(\boldsymbol{x}_k + \nu_k \boldsymbol{e}_j) - f^i(\boldsymbol{x}_k)}{\nu_k} \boldsymbol{e}_j \tag{2.17}$$

where $G^i(\boldsymbol{x}_k, \nu_k)$ is the gradient estimator for function $f^i(\boldsymbol{x}_k)$, ν_k is the difference length, $\boldsymbol{e}_j$ is the j-th coordinator vector.

As the gradient estimator is not identical to the true gradient, the estimation deviation between the estimator and true gradient exists. Though in many applications, the estimated gradients are accurate enough to provide a good solution, in safety-critical optimization, however, the deviation of gradient estimator from the true one shall be considered with much cares, since an inaccurate gradient information may lead to safety violation in function evaluations.

Denote by Δ_k^i the difference between the gradient estimator $G^i(\boldsymbol{x}_k, \nu_k)$ and the true gradient $\nabla f^i(\boldsymbol{x}_k)$:

$$\Delta_k^i = G^i(\boldsymbol{x}_k, \nu_k) - \nabla f^i(\boldsymbol{x}_k) \tag{2.18}$$

With Assumption 1, the following holds:

Lemma 1 ([UKK19]). *The norm of estimation deviation Δ_k^i (2.18) is upper bounded by:*

$$\|\Delta_k^i\|_2 = \|G^i(\boldsymbol{x}_k, \nu_k) - \nabla f^i(\boldsymbol{x}_k)\|_2 \leq \frac{\sqrt{d}M\nu_k}{2} \tag{2.19}$$

Proof. *According to the assumption 1 of M-smooth on function $f^i(\boldsymbol{x})$, there is:*

$$f^i(\boldsymbol{x}_k + \nu_k \boldsymbol{e}_j) \leq f^i(\boldsymbol{x}_k) + \nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}(\nu_k \boldsymbol{e}_j) + \frac{1}{2}M\|\nu_k \boldsymbol{e}_j\|_2^2 \tag{2.20}$$

using (2.17) and (2.18), we have:

$$\|\Delta_k^i\|_2 = \sqrt{\sum_{j=1}^{d}\left[\frac{f^i(\boldsymbol{x}_k + \nu_k \boldsymbol{e}_j) - f^i(\boldsymbol{x}_k)}{\nu_k} - \nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{e}_j\right]^2 \|\boldsymbol{e}_j\|_2^2} \tag{2.21}$$

$$= \sqrt{\sum_{j=1}^{d}\left[\frac{f^i(\boldsymbol{x}_k + \nu_k \boldsymbol{e}_j) - f^i(\boldsymbol{x}_k) - \nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}(\nu_k \boldsymbol{e}_j)}{\nu_k}\right]^2} \tag{2.22}$$

$$\leq \sqrt{\sum_{j=1}^{d}\left[\frac{M\|\nu_k \boldsymbol{e}_j\|_2^2}{2\nu_k}\right]^2} \tag{2.23}$$

$$= \frac{\sqrt{d}M\nu_k}{2} \tag{2.24}$$

□

It can be seen that the estimation error $\|\Delta_k^i\|_2$ depends on the step length ν_k, the dimension of function d and the smoothness constant M. As for a certain application, the dimension d is fixed, and the smoothness property is determined by the underlying function. The estimation error can be handled by adapting the step length ν_k.

Given an acceptable upper bound μ on the gradient estimation deviation $\|\Delta_k^i\|_2$:

$$\|\Delta_k^i\|_2 \leq \mu \tag{2.25}$$

the step length satisfying the estimation accuracy can be computed by:

$$\nu_k = \frac{2\mu}{\sqrt{d}M} \tag{2.26}$$

In addition to the gradient estimation accuracy, the safety requirement shall be fulfilled while estimating the gradient. That is, the chosen step length ν_k is such that the function evaluations at $\boldsymbol{x}_k + \nu_k \boldsymbol{e}_j$ do not violate the safety constraints.

Given the *Lipschitz* assumption 1 about the constraint functions $f^i(\boldsymbol{x})$, a safety guaranteed step length can be computed by the following.

Lemma 2 ([UKK19]). *For* $\nu_k \leq \frac{\min\limits_{i=1,\ldots,m}\left\{-f^i(\boldsymbol{x}_k)\right\}}{2L}$, *the evaluations made around feasible point at* $\boldsymbol{x}_k + \nu_k \boldsymbol{e}_j$ *are safety feasible.*

Proof. *For any point* $\boldsymbol{y} \in \mathbb{R}^d$ *satisfying* $\|\boldsymbol{y} - \boldsymbol{x}_k\|_2 \leq \frac{\min\limits_{i=1,\ldots,m}\left\{-f^i(\boldsymbol{x}_k)\right\}}{2L}$, *there is:*

$$
\begin{aligned}
f^i(\boldsymbol{y}) &\leq f^i(\boldsymbol{x}_k) + \nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{x}_k) && (2.27)\\
&\leq f^i(\boldsymbol{x}_k) + L\|(\boldsymbol{y} - \boldsymbol{x}_k)\|_2 && (2.28)\\
&\leq f^i(\boldsymbol{x}_k) + L\frac{-f^i(\boldsymbol{x}_k)}{2L} && (2.29)\\
&\leq \frac{1}{2} f^i(\boldsymbol{x}_k) \leq 0 && (2.30)
\end{aligned}
$$

□

Considering both the estimation accuracy and safety constraints, the step length ν_k for computing the gradient estimator is chosen as:

$$
\nu_k = \min\left\{ \frac{2\mu}{\sqrt{d}M}, \frac{\min\limits_{i=1,\ldots,m}\{-f^i(\boldsymbol{x}_k)\}}{2L} \right\} \tag{2.31}
$$

With the computed gradient estimator $G^i(\boldsymbol{x}_k, \nu_k)$, the search direction for optimization can be determined.

2.5.3 Search direction

The search direction is the direction along which the objective function is decreased. It can be determined according to the gradient of the objective function. With the gradient information, there are two common search directions:

- The steepest direction;
- The *Newton direction*.

The steepest direction is the most obvious search direction, which is defined as the direction of negative gradient of the objective function. Among all directions one can move from $\boldsymbol{x}_k$, the steepest direction is the one along which the objective function decreases most rapidly. Let $\boldsymbol{p}_k \in \mathbb{R}^d$ be a search direction. The steepest direction can be written as:

$$
\boldsymbol{p}_k^s = -\nabla f^0(\boldsymbol{x}_k) \tag{2.32}
$$

where $\nabla f^0(\boldsymbol{x}_k)$ is the gradient of the objective function at point $\boldsymbol{x}_k$.

Though the steepest direction has an advantage that it requires only the first

order gradient of the objective function, it may be slow for difficult problems.

The *Newton direction* considers the second-order information of the objective function. It is derived from the second-order Taylor's theorem 1. If $\boldsymbol{p}_k$ is a search direction at point $\boldsymbol{x}_k$, then the function value at $\boldsymbol{x}_k + \boldsymbol{p}_k$ can be estimated by:

$$f^0(\boldsymbol{x}_k + \boldsymbol{p}_k) \approx f^0(\boldsymbol{x}_k) + \nabla f^0(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{p}_k + \frac{1}{2} \boldsymbol{p}_k^{\mathrm{T}} \nabla^2 f^0(\boldsymbol{x}_k) \boldsymbol{p}_k \tag{2.33}$$

If the Hessian $\nabla^2 f^0(\boldsymbol{x}_k)$ at $\boldsymbol{x}_k$ is positive definite, the *Newton direction* is defined as the vector $\boldsymbol{p}_k$ that minimizes function $f^i(\boldsymbol{x}_k + \boldsymbol{p}_k)$. The solution of this problem can be computed by setting the derivative of $f^0(\boldsymbol{x}_k + \boldsymbol{p}_k)$ to zero. By this computation, the *Newton direction* $\boldsymbol{p}_k^N$ is defined as:

$$\boldsymbol{p}_k^N = - \left(\nabla^2 f^0(\boldsymbol{x}_k) \right)^{-1} \nabla f^0(\boldsymbol{x}_k) \tag{2.34}$$

The *Newton direction* has an advantage of fast local convergence. Once reaching to a neighborhood of the solution, the optimal point can be found in only a few iterations. The main drawback of *Newton direction* is the requirement of Hessian $\nabla^2 f^0(\boldsymbol{x}_k)$ and its positive definite. In model-free optimization problem, the estimation of second-order gradients can be computationally expensive and sensitive to measurement errors.

An attractive alternative to *Newton direction* is the *Quasi-Newton direction*. It maintains the superlinear convergence property of the *Newton direction* while does not require computation of Hessian. The *Quasi-Newton direction* replaces $\nabla^2 f^0(\boldsymbol{x}_k)$ by an approximated Hessian matrix $\boldsymbol{Y}_k$, and iteratively updates the approximation using information obtained from previous iterations.

The idea behind the Hessian approximation is that the changes in the gradient $\nabla f^0(\boldsymbol{x}_k)$ provides information about the second-order derivative of the objective function.

By the definition of second-order gradient, we know that:

$$\nabla^2 f^0(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \approx \nabla f^0(\boldsymbol{x}_k) - \nabla f^0(\boldsymbol{x}_{k+1}) \tag{2.35}$$

The approximated Hessian matrix $\boldsymbol{Y}_{k+1}$ satisfies Equation (2.35) as:

$$\boldsymbol{Y}_{k+1} \boldsymbol{s}_k = \boldsymbol{g}_k \tag{2.36}$$

where

$$\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \tag{2.37}$$

$$\boldsymbol{g}_k = \nabla f^0(\boldsymbol{x}_{k+1}) - \nabla f^0(\boldsymbol{x}_k) \tag{2.38}$$

For updating the Hessian matrix, one popular formula is the *BFGS formula*, which is

$$\boldsymbol{Y}_{k+1} = \boldsymbol{Y}_k - \frac{\boldsymbol{Y}_k \boldsymbol{s}_k \boldsymbol{s}_k^{\mathrm{T}} \boldsymbol{Y}_k}{\boldsymbol{s}_k^{\mathrm{T}} \boldsymbol{Y}_k \boldsymbol{s}_k} + \frac{\boldsymbol{g}_k \boldsymbol{g}_k^{\mathrm{T}}}{\boldsymbol{g}_k^{\mathrm{T}} \boldsymbol{s}_k} \tag{2.39}$$

A good property of *BFGS update* is that it maintains the positive definite as long as the initial matrix $\boldsymbol{Y}_0$ is positive definite and $\boldsymbol{s}_k^{\mathrm{T}\boldsymbol{g}_k} > 0$.

In practical implementation, to reduce the computation cost involved in computing the inverse of Hessian in (2.34), the update can be done directly on the inverse Hessian. Let $\boldsymbol{H}_k := \boldsymbol{Y}_k^{-1}$, the update rule for $\boldsymbol{H}_k$ is:

$$\boldsymbol{H}_{k+1} = \left(\boldsymbol{I} - \rho_k \boldsymbol{s}_k \boldsymbol{g}_k^{\mathrm{T}}\right) \boldsymbol{H}_k \left(\boldsymbol{I} - \rho_k \boldsymbol{g}_k \boldsymbol{s}_k^{\mathrm{T}}\right) + \rho_k \boldsymbol{s}_k \boldsymbol{s}_k^{\mathrm{T}} \tag{2.40}$$

where

$$\rho_k = \frac{1}{\boldsymbol{g}_k^{\mathrm{T}} \boldsymbol{s}_k} \tag{2.41}$$

Then the *Quasi-Newton* search direction with approximated Hessian matrix is:

$$\boldsymbol{p}_k^N = -\boldsymbol{H}_k \nabla f^0(\boldsymbol{x}_k) \tag{2.42}$$

2.5.4 Safe set formulation

During the optimization process, the safety requirement shall be fulfilled at each iteration. To ensure this requirement, a local safe set $\mathcal{S}_k \in \mathcal{D}$ is formulated at each iteration, so that the function evaluations inside this safe set do not violate constraints.

If a current safe point $\boldsymbol{x}_k$ is given, the safety requirement throughout the optimization process is equivalent to guarantee that moving from a safe point $\boldsymbol{x}_k$, the next point $\boldsymbol{x}_{k+1}$ does not violate constraints. That is, $f^i(\boldsymbol{x}_{k+1}) \leq 0$ for $\forall i = 1, \ldots, m$. A local safe set $\mathcal{S}_k^i$ for constraint $f^i(\boldsymbol{x})$ at point $\boldsymbol{x}_k$ is defined as the set in which all point $\boldsymbol{x}_{k+1}$ satisfy the safety constraints.

Given the function value at the current point $f^i(\boldsymbol{x}_k)$, with the smoothness assumption 1 about the unknown function, using the Taylor's theorem 1, the function value at the next point can be upper bounded by:

$$f^i(\boldsymbol{x}_{k+1}) \leq f^i(\boldsymbol{x}_k) + \nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \frac{1}{2} M \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \tag{2.43}$$

By forcing the upper bound less than zero, the safe set for $\boldsymbol{x}_{k+1}$ can be computed by:

$$\mathcal{S}_k^i := \{\boldsymbol{x} \in \mathbb{R}^d : f^i(\boldsymbol{x}_k) + \nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2} M \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 \leq 0\} \tag{2.44}$$

Since in model-free optimization problem, the true gradient $\nabla f^i(\boldsymbol{x}_k)$ is not available, it is replaced by a gradient estimator $G^i(\boldsymbol{x}_k, \nu_k)$. To alleviate the effect of estimation deviation $\|\Delta_k^i\|_2$ on the safety properties of $\mathcal{S}_k^i$, some manipulations on the gradient estimator shall be made so that the computed safe set does not lost safety guarantee for the sake of estimation deviation. By the following manipulation, the safety of $\mathcal{S}_k^i$ is maintained in presence of estimation deviation.

Theorem 3. *Let $\boldsymbol{p}_k$ be the search direction, $G^i(\boldsymbol{x}_k, \nu_k)$ be the gradient estimator and $\|\Delta_k^i\|_2$ be the estimation deviation from $\nabla f^i(\boldsymbol{x}_k)$, then all points $\boldsymbol{x}_{k+1}$ satisfying the following constraints are feasible.*

$$f^i(\boldsymbol{x}_k) + \hat{G}^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \frac{1}{2}M\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \leq 0 \tag{2.45}$$

where

$$\hat{G}^i(\boldsymbol{x}_k, \nu_k) = G^i(\boldsymbol{x}_k, \nu_k) + \frac{\sqrt{d}M\nu_k\|\boldsymbol{p}_k\|_2}{2\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_k}\boldsymbol{e} \tag{2.46}$$

with a unit coordinate vector $\boldsymbol{e}$.

Proof. *Recall that $\boldsymbol{x}_{k+1}$ is feasible for constraint $f^i(\boldsymbol{x})$, if (2.43) holds. To prove Theorem 3, it is equivalent to prove that with (2.45), (2.43) always holds. Let α be the angle between $\nabla f^i(\boldsymbol{x}_k)$ and $\boldsymbol{p}_k$ and β be the angle between $G^i(\boldsymbol{x}_k, \nu_k)$ and $\boldsymbol{p}_k$. With (2.45), (2.43) holds if the following is true:*

$$\nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \leq \hat{G}^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \tag{2.47}$$

let $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = \alpha\boldsymbol{p}_k$, then (2.47) is equivalent to:

$$\alpha\nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{p}_k \leq \alpha\hat{G}^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}\boldsymbol{p}_k \tag{2.48}$$

$$\nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{p}_k \leq \hat{G}^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}\boldsymbol{p}_k \tag{2.49}$$

$$\nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{p}_k \leq G^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}\boldsymbol{p}_k + \frac{\sqrt{d}M\nu_k\|\boldsymbol{p}_k\|_2}{2\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_k}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_k \tag{2.50}$$

$$\|\nabla f^i(\boldsymbol{x}_k)\|_2\|\boldsymbol{p}_k\|_2\cos\alpha \leq \|G^i(\boldsymbol{x}_k, \nu_k)\|_2\|\boldsymbol{p}_k\|_2\cos\beta + \frac{\sqrt{d}M\nu_k\|\boldsymbol{p}_k\|_2}{2} \tag{2.51}$$

$$\|\nabla f^i(\boldsymbol{x}_k)\|_2\cos\alpha - \|G^i(\boldsymbol{x}_k, \nu_k)\|_2\cos\beta \leq \frac{\sqrt{d}M\nu_k}{2} \tag{2.52}$$

Let ψ be the angle between $G^i(\boldsymbol{x}_k, \nu_k) - \nabla f^i(\boldsymbol{x}_k)$ and $-\boldsymbol{p}_k$. Since $G^i(\boldsymbol{x}_k, \nu_k)$ is either inside or on the surface of a ball $\mathcal{B}\left(\nabla f^i(\boldsymbol{x}_k), \|\Delta_k^i\|_2\right)$ centered at $\nabla f^i(\boldsymbol{x}_k)$ with radius $\|\Delta_k^i\|_2$ as shown in Figure 2.1, there is:

$$\left(\|\nabla f^i(\boldsymbol{x}_k)\|_2\cos\alpha - \|G^i(\boldsymbol{x}_k, \nu_k)\|_2\cos\beta\right)^2 = \left(\|\Delta_k^i\|_2\cos\psi\right)^2 \tag{2.53}$$

$$\left|\|\nabla f^i(\boldsymbol{x}_k)\|_2\cos\alpha - \|G^i(\boldsymbol{x}_k, \nu_k)\|_2\cos\beta\right| = \left|\|\Delta_k^i\|_2\cos\psi\right| \tag{2.54}$$

$$\|\nabla f^i(\boldsymbol{x}_k)\|_2\cos\alpha - \|G^i(\boldsymbol{x}_k, \nu_k)\|_2\cos\beta \leq \|\Delta_k^i\|_2 \leq \frac{\sqrt{d}M\nu_k}{2} \tag{2.55}$$

Thus (2.43) always holds, and $\boldsymbol{x}_{k+1}$ is feasible for constraint $f^i(\boldsymbol{x})$. □

With Theorem 3, a conservative safe set $\hat{\mathcal{S}}_k^i$ for constraint $f^i(\boldsymbol{x})$ is defined as:

$$\hat{\mathcal{S}}_k^i := \{\boldsymbol{x} \in \mathbb{R}^d : f^i(\boldsymbol{x}_k) + \hat{G}^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}M\|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 \leq 0\} \tag{2.56}$$

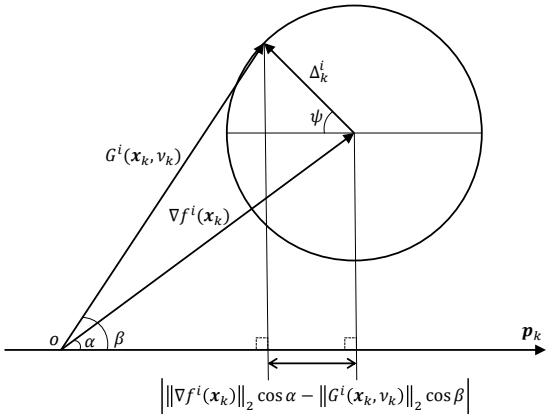


Figure 2.1: $G^i(\boldsymbol{x}_k, \nu_k)$ is inside or on the surface of a ball $\mathcal{B}\left(\nabla f^i(\boldsymbol{x}_k), \|\Delta_k^i\|_2\right)$

Lemma 3. *Safe set $\hat{\mathcal{S}}_k^i$ is a ball $\mathcal{B}$ with center $\hat{\mathcal{O}}_k^i$ and radius $\hat{\mathcal{R}}_k^i$:*

$$\hat{\mathcal{O}}_k^i = \boldsymbol{x}_k - \frac{1}{M}\hat{G}^i(\boldsymbol{x}_k, \nu_k) \tag{2.57}$$

$$\hat{\mathcal{R}}_k^i = \sqrt{\frac{1}{M}\left(\frac{\|\hat{G}^i(\boldsymbol{x}_k, \nu_k)\|_2^2}{M} - 2f^i(\boldsymbol{x}_k)\right)} \tag{2.58}$$

Proof. *Equation (2.45) can be rewritten as:*

$$\frac{1}{2}M\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 + \hat{G}^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + f^i(\boldsymbol{x}_k) \leq 0 \tag{2.59}$$

$$\frac{1}{2}M\left(\|\boldsymbol{x}_{k+1}\|_2^2 + \|\boldsymbol{x}_k\|_2^2 - 2\boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{x}_{k+1}\right) + \hat{G}^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + f^i(\boldsymbol{x}_k) \leq 0 \tag{2.60}$$

$$\frac{1}{2}M\|\boldsymbol{x}_{k+1}\|_2^2 + \left(\hat{G}^i(\boldsymbol{x}_k, \nu_k) - M\boldsymbol{x}_k\right)^{\mathrm{T}}\boldsymbol{x} + \frac{1}{2}M\|\boldsymbol{x}_k\|_2^2 - \hat{G}^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}\boldsymbol{x}_k + f^i(\boldsymbol{x}_k) \leq 0 \tag{2.61}$$

$$\|\boldsymbol{x}_{k+1}\|_2^2 + \underbrace{\frac{2}{M}\left(\hat{G}^i(\boldsymbol{x}_k, \nu_k) - M\boldsymbol{x}_k\right)^{\mathrm{T}}}_{P}\boldsymbol{x} + \underbrace{\|\boldsymbol{x}_k\|_2^2 - \frac{2}{M}\hat{G}^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}\boldsymbol{x}_k + \frac{2}{M}f^i(\boldsymbol{x}_k)}_{Q} \leq 0 \tag{2.62}$$

then, the center is

$$\hat{\mathcal{O}}_k^i = -\frac{1}{2}\cdot P \tag{2.63}$$

$$= -\frac{1}{2}\cdot\frac{2}{M}\left(\hat{G}^i(\boldsymbol{x}_k, \nu_k) - M\boldsymbol{x}_k\right) \tag{2.64}$$

$$= \boldsymbol{x}_k - \frac{1}{M}\hat{G}^i(\boldsymbol{x}_k, \nu_k) \tag{2.65}$$

and the radius is

$$\hat{\mathcal{R}}_k^i = \sqrt{\frac{1}{4}(\|P\|_2^2 - 4Q)} \tag{2.66}$$

$$= \sqrt{\frac{1}{4}\left[\left\|\frac{2}{M}(\hat{G}^i(\boldsymbol{x}_k, \nu_k) - M\boldsymbol{x}_k)\right\|_2^2 - 4\left[\|\boldsymbol{x}_k\|_2^2 - \frac{2}{M}\hat{G}^i(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}}\boldsymbol{x}_k + \frac{2}{M}f^i(\boldsymbol{x}_k)\right]\right]} \tag{2.67}$$

$$= \sqrt{\frac{1}{M^2}\left(\|\hat{G}^i(\boldsymbol{x}_k, \nu_k)\|_2^2 + M^2\|\boldsymbol{x}_k\|_2^2\right) - \|\boldsymbol{x}_k\|_2^2 - \frac{2}{M}f^i(\boldsymbol{x}_k)} \tag{2.68}$$

$$= \sqrt{\frac{1}{M}\left(\frac{\|\hat{G}^i(\boldsymbol{x}_k, \nu_k)\|_2^2}{M} - 2f^i(\boldsymbol{x}_k)\right)} \tag{2.69}$$

□

Since the safety shall be satisfied for all constraints $f^i(\boldsymbol{x})$, $\forall i = 1, \ldots, m$, the overall safe set $\mathcal{S}_k$ at iteration k is the intersection of all $\hat{\mathcal{S}}_k^i$:

$$\mathcal{S}_k := \cap_{i=1}^m \hat{\mathcal{S}}_k^i \tag{2.70}$$

Using Lemma 3, safe set (2.70) is equivalent to:

$$\mathcal{S}_k := \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x} - \hat{\mathcal{O}}_k^i\|_2 \leq \hat{\mathcal{R}}_k^i,\ \forall i = 1, \ldots, m\} \tag{2.71}$$

With the computed safe set, the function evaluations can be made inside the safe set without violating constraints. The next step is to find a step length along the search direction inside the safe set, such that the objective function decreases rapidly.

2.5.5 Search direction projection

In constrained optimization problem, if the optimal point of the objective function is outside of the feasible set $\mathcal{D}$, the search direction shall be projected onto the normal vector of the surface spanned by gradients of all active constraints ($f^i(\boldsymbol{x}_k) = 0$). Since, if a current point is on a boundary of the feasible set, and the search direction is not parallel to the gradient of the active constraint and the angle between the search direction and the active constraint's gradient is less than $\frac{\pi}{2}$, then there exist a direction perpendicular to the constraint's gradient such that along this search direction, the objective can be reduced while not violating the constraint.

Let us assume that the search direction $\boldsymbol{p}_k$ is the negative gradient of the objective function $-\nabla f^0(\boldsymbol{x}_k)$, an illustration of the search direction projection can be seen in Figure 2.2.

Lemma 4. *A point is a KKT point of the constrained optimization problem 2, if at this point, the gradient of the objective function is a linear combination of gradients of all active constraints with non-negative coefficients.*

Proof. *Let $\mathcal{A}$ be the set of indices of all active constraints at point $\boldsymbol{x}_k$, set all Lagrangian multipliers for inactive constraints to zeros:*

$$f^i(\boldsymbol{x}_k) < 0,\ i \notin \mathcal{A} \tag{2.72}$$

$$\lambda_k^i = 0,\ i \notin \mathcal{A} \tag{2.73}$$

If there exists a set of non-negative multipliers for active constraints such that the following holds:

$$-\nabla f^0(\boldsymbol{x}_k) = \sum_{i \in \mathcal{A}} \lambda_k^i \nabla f^i(\boldsymbol{x}_k) \tag{2.74}$$

$$f^i(\boldsymbol{x}_k) = 0,\ i \in \mathcal{A} \tag{2.75}$$

$$\lambda_k^i \geq 0,\ i \in \mathcal{A} \tag{2.76}$$

Then, the derivative of Lagrangian $\mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$ 2.11 can be written as:

$$\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) = \nabla f^0(\boldsymbol{x}_k) + \sum_{i \in \mathcal{A}} \lambda_k^i \nabla f^i(\boldsymbol{x}_k) + \sum_{i \notin \mathcal{A}} \lambda_k^i \nabla f^i(\boldsymbol{x}_k) = 0 \tag{2.77}$$

and there is:

$$f^i(\boldsymbol{x}_k) \leq 0,\ i = 1, \ldots, m \tag{2.78}$$

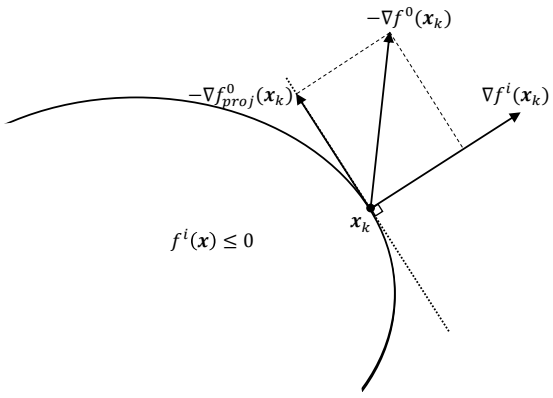


Figure 2.2: Search direction $\boldsymbol{p}_k = -\nabla f^0(\boldsymbol{x}_k)$ projected onto the normal vector of constraint's gradient $\nabla f^i(\boldsymbol{x}_k)$

$$\lambda_k^i \geq 0,\ i = 1, \ldots, m \tag{2.79}$$
$$\lambda_k^i f^i(\boldsymbol{x}_k) = 0,\ i = 1, \ldots, m \tag{2.80}$$

Thus, the KKT condition 2 is satisfied and the point $\boldsymbol{x}_k$ is a KKT point. □

Geometrically, (2.74) means that the negative gradient of objective function is inside a convex cone spanned by all active constraint's gradients.
The search direction projection $\boldsymbol{p}_k^{proj}$ can be found by the following:

Lemma 5. *Let $\boldsymbol{C}_k \in \mathbb{R}^{d \times |\mathcal{A}|}$ be the the matrix with columns of gradients of all active constraints $\nabla f^i(\boldsymbol{x}_k)$, $i \in \mathcal{A}$, the minimal angle between $-\nabla f^0(\boldsymbol{x}_k)$ and $f^i(\boldsymbol{x}_k)$ is less than $\frac{\pi}{2}$, $\boldsymbol{\lambda}_{sol} \in \mathbb{R}^{|\mathcal{A}|}$ be the solution vector of the following non-negative linear least-squares problem:*

Problem 3. ***Non-Negative Linear Least Squares Problem (Nonneglsp)***

$$\underset{\boldsymbol{\lambda} \in \mathbb{R}^{|\mathcal{A}|}}{\text{minimize}} \quad \|\boldsymbol{C}_k \boldsymbol{\lambda} - \boldsymbol{p}_k\|_2^2 \tag{2.81}$$
$$\text{subject to} \quad \lambda^i \geq 0,\ i \in \mathcal{A} \tag{2.82}$$

Direction $\boldsymbol{p}_k^{proj}$ is a descent direction and all active constraints do not increase along this direction, if $\boldsymbol{x}_k$ is not a KKT point and:

$$\boldsymbol{p}_k^{proj} = \boldsymbol{p}_k - \boldsymbol{p}_k' \tag{2.83}$$

where $\boldsymbol{p}_k$ is the negative gradient of the objective function:

$$\boldsymbol{p}_k = -\nabla f^0(\boldsymbol{x}_k) \tag{2.84}$$

and $\boldsymbol{p}_k'$ is:

$$\boldsymbol{p}_k' = \boldsymbol{C}_k \boldsymbol{\lambda}_{sol} \tag{2.85}$$

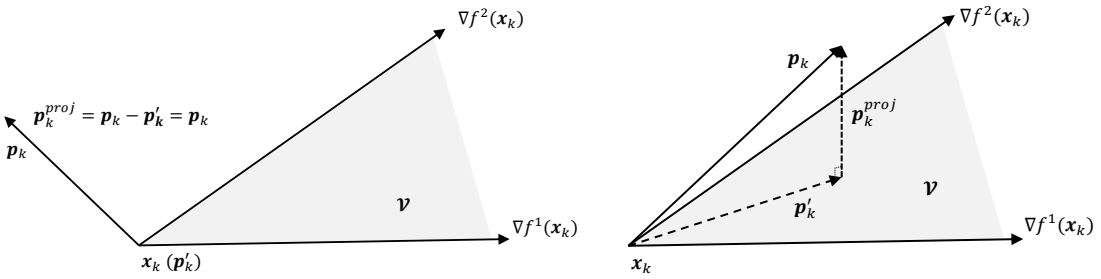
Proof. *Let J_{min} be the minimum of (2.81) at point $\boldsymbol{x}_k$. If $J_{min} = 0$, then $\boldsymbol{\lambda}_{sol}$ is a non-negative multiplier vector such that the following holds:*

$$-\nabla f^0(\boldsymbol{x}_k) = \boldsymbol{C}_k \boldsymbol{\lambda}_{sol} = \sum_{i \in \mathcal{A}} \lambda_{sol}^i \nabla f^i(\boldsymbol{x}_k) \tag{2.86}$$

According to Lemma 4, the point $\boldsymbol{x}_k$ is a KKT point.

If $J_{min} > 0$, then $-\nabla f^0(\boldsymbol{x}_k)$ can not be exactly expressed by a linear combination of gradients of all active constraints, thus, $\boldsymbol{x}_k$ is not a KKT point.

Let φ be the angle between $\boldsymbol{p}_k$ and an arbitrary vector $\boldsymbol{p}_k^a$ in the convex



(a) $\boldsymbol{p}_k^{proj}$ when $\varphi_{min} \geq \frac{\pi}{2}$ (b) $\boldsymbol{p}_k^{proj}$ when $\varphi_{min} < \frac{\pi}{2}$

Figure 2.3: Search direction projection $\boldsymbol{p}_k^{proj}$ in $\mathbb{R}^3$

cone $\mathcal{V}$ spanned by $\nabla f^i(\boldsymbol{x}_k),\ i \in \mathcal{A}$, γ be the angle between any two vectors of $\nabla f^i(\boldsymbol{x}_k),\ i \in \mathcal{A}$:

$$\varphi = \measuredangle\left(\boldsymbol{p}_k,\ \boldsymbol{p}_k^a\right),\ \forall \boldsymbol{p}_k^a \in \mathcal{V} \tag{2.87}$$

$$\gamma = \measuredangle\left[\nabla f^i(\boldsymbol{x}_k),\ \nabla f^j(\boldsymbol{x}_k)\right],\ i, j \in \mathcal{A} \tag{2.88}$$

One can see that:

$$J_{min} = \min \left\|\boldsymbol{C}_k \boldsymbol{\lambda}_{sol} - \left(-\nabla f^0(\boldsymbol{x}_k)\right)\right\|_2^2 = \min \left\|\boldsymbol{p}'_k - \boldsymbol{p}_k\right\|_2^2 \tag{2.89}$$

which means that $\boldsymbol{p}'_k$ is the best approximation of $\boldsymbol{p}_k$ among all possible $\boldsymbol{p}_k^a$ in the convex cone $\mathcal{V}$. Geometrically, it is equivalent to that the distance between $\boldsymbol{p}'_k$ and $\boldsymbol{p}_k$ is minimal among all possible distances between $\boldsymbol{p}_k$ and $\boldsymbol{p}_k^a$, and thus $\boldsymbol{p}_k^{proj} = \boldsymbol{p}_k - \boldsymbol{p}'_k$ is perpendicular to $\boldsymbol{p}'_k$ and the minimal possible angle φ_{min} is the angle between $\boldsymbol{p}_k$ and $\boldsymbol{p}'_k$.

$$\varphi_{min} = \measuredangle(\boldsymbol{p}_k,\ \boldsymbol{p}'_k) \tag{2.90}$$

If $\varphi_{min} \geq \frac{\pi}{2}$, then there is $\boldsymbol{\lambda}_{sol} = \boldsymbol{0}$ and $\|\boldsymbol{p}'_k\|_2 = 0$, which means that $\boldsymbol{p}'_k$ is a point located at $\boldsymbol{x}_k$, and thus, $\boldsymbol{p}_k^{proj} = \boldsymbol{p}_k - \boldsymbol{p}'_k = \boldsymbol{p}_k$. Since $\boldsymbol{p}_k = -\nabla f^0(\boldsymbol{x}_k)$, thus $\boldsymbol{p}_k^{proj}$ is a descent direction.

if $\varphi_{min} < \frac{\pi}{2}$, then there is:

$$\measuredangle(\boldsymbol{p}_k,\ \boldsymbol{p}_k^{proj}) = \frac{\pi}{2} - \measuredangle(\boldsymbol{p}_k,\ \boldsymbol{p}'_k) = \frac{\pi}{2} - \varphi_{min} \leq \frac{\pi}{2} \tag{2.91}$$

Thus, $\boldsymbol{p}_k^{proj}$ is a descent direction. An illustration of search direction projection in $\mathbb{R}^3$ is shown in Figure 2.3.

As one knows that in a convex cone $\mathcal{V}$, the maximum possible angle γ_{max} between any two vectors of $\nabla f^i(\boldsymbol{x}_k),\ i \in \mathcal{A}$ satisfies $\gamma_{max} \leq \pi$ (otherwise $\mathcal{V}$ is not convex), hence, the angle between $\boldsymbol{p}_k^{proj}$ and any vector $\boldsymbol{p}_k^a$ in $\mathcal{V}$ satisfies:

$$\frac{\pi}{2} \leq \measuredangle\left(\boldsymbol{p}_k^{proj}, \boldsymbol{p}_k^a\right) \leq \frac{\pi}{2} + \gamma_{max} \leq \frac{3\pi}{2} \tag{2.92}$$

Thus, along the direction $\boldsymbol{p}_k^{proj}$, all active constraints do not increase. □

Problem 3 can be solved using non-negative least-square solvers such as the *lsqnonneg* function in numerical analysis software MATLAB.

The search direction projection is computed until J_{min} converges to zero. The decision point in this case is a KKT point of the constrained optimization problem 2. In practice, this condition is satisfied if J_{min} is less than some tolerance value ϵ, e.g. $\epsilon = 10^{-10}$.

2.5.6 Step length computation

The choice of step length shall have the following properties: first, it provides a sufficient decrease in the objective function; second, the function evaluation at the next step shall not violate the safety constraints.

Ideally, with the selected step length, the next function evaluation shall have the minimum possible value along this search direction, while satisfies safety constraints. Denote by α_k the step length at iteration k, finding a step length with above properties is equivalent to solve the following optimization problem:

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & f^0(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k) \\ \text{subject to} \quad & \boldsymbol{x}_k + \alpha \boldsymbol{p}_k \in \mathcal{S}_k \\ & \alpha_k > 0 \end{aligned} \tag{2.93}$$

Solving this problem aims to find a step length that minimizes the objective function at the next iteration while satisfying safety constraints. However, it is generally expensive to find a global minimizer of this problem, as there may be many local optimums along the search direction. A practical way is to find an inexact solution of the problem, with which the objective decreases not by the maximum possible value, instead, by only a good enough amount, and then iteatively improve the solution. An inexact solution of the step length can be found using the *line search* algorithm. In the algorithm, it tries a set of candidate step lengths and selects the one that provides enough decrease in the objective function.

A simple condition for selection of candidate step length is:

$$f^0(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) < f^0(\boldsymbol{x}_k) \tag{2.94}$$

which means that the step length is acceptable as long as the objective at the next iteration is less than the current. However, this condition may lead to a poor convergence performance, as any little reduction in the objective function is accepted. In order to provide a better convergence result, a *sufficient decrease condition* is applied. By this condition, only the step length with which the objective decreases by enough amount are selected. The *Armijo condition* states such a measurement, which is:

$$f^0(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) < f^0(\boldsymbol{x}_k) + c\alpha_k \nabla f^0(\boldsymbol{x}_k)^{\mathrm{T}} \boldsymbol{p}_k \tag{2.95}$$

where $c \in (0,1)$ is a constant that measures the decrease level. The *Armijo condition* implies that with the step length α_k along the search direction $\boldsymbol{p}_k$, the decrease in the objective function are larger than $\|c\alpha_k \nabla f^0(\boldsymbol{x}_k)\boldsymbol{p}_k\|_2$.

To illustrate the effect of constant c, let $l(\alpha)$, $h(\alpha)$ be:

$$l(\alpha) = f^0(\boldsymbol{x}_k) + c\alpha_k \nabla f^0(\boldsymbol{x}_k)^{\mathrm{T}} \boldsymbol{p}_k \tag{2.96}$$

$$h(\alpha) = f^0(\boldsymbol{x}_k) + \alpha_k \nabla f^0(\boldsymbol{x}_k)^{\mathrm{T}} \boldsymbol{p}_k \tag{2.97}$$

the *Armijo condition* can be interpreted by Figure 2.4. It can be seen that with

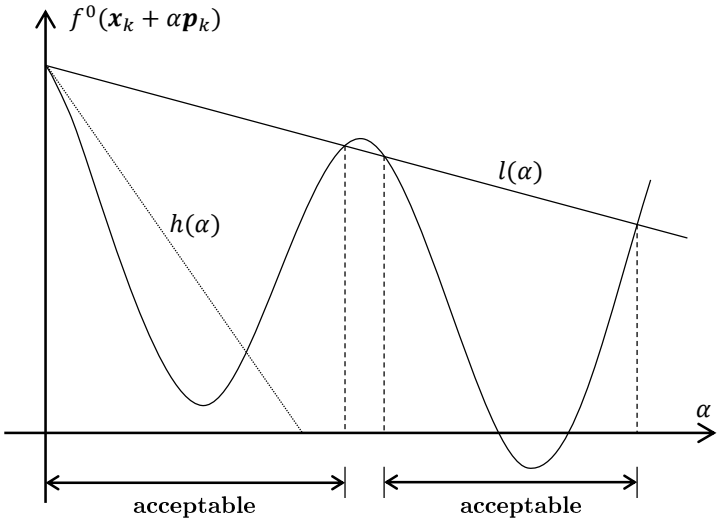


Figure 2.4: Acceptable step length

different constant c, the range of acceptable step lengths varies. In practice, c is often chosen to be a small value to incorporate a larger range. In model-free optimization problem, the exact gradient $\nabla f^0(\boldsymbol{x}_k)$ is replaced by a gradient estimator $G^0(\boldsymbol{x}_k, \nu_k)$.

In addition to the sufficient decrease requirement, in safe optimization problem, the safety requirement shall be satisfied. As the line search approach tries a sequence of candidate step lengths, a trial of too large step length may lead to violation of the safety constraints. To overcome this problem, an upper bound on the possible candidate step lengths are defined, within which all candidate step lengths satisfy the safety constraints. The upper bound of candidate step lengths is such that all trial points used when searching for the step length are inside the safe set $\mathcal{S}_k$. Using (2.71), the upper bound of candidate step lengths $\hat{\alpha}_k$ along the search direction $\boldsymbol{p}_k$ is defined as:

$$\hat{\alpha}_k = \max\{\alpha \in \mathbb{R} : \|\boldsymbol{x}_k + \alpha \boldsymbol{p}_k - \hat{\mathcal{O}}_k^i\|_2 < \hat{\mathcal{R}}_k^i, \forall i = 1, \ldots, m, \alpha > 0\} \tag{2.98}$$

The step length upper bound is illustrated in Figure 2.5.

Besides the upper bound on candidate step length, during testing step lengths,

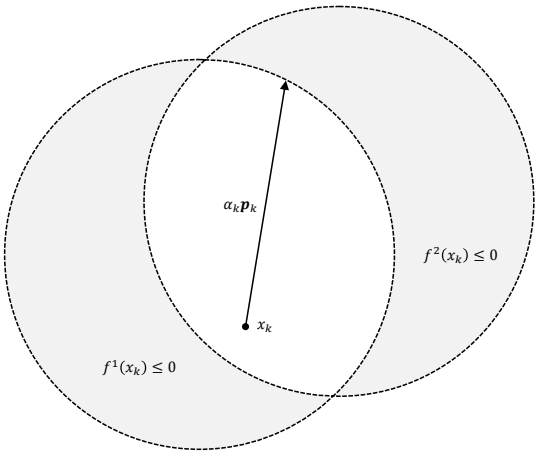


Figure 2.5: Step length upper bound

an additional safety condition shall be satisfied by acceptable step length. Since in model-free optimization, the gradient estimators and function measurements involve uncertainties, the safety constraints $f^i(\boldsymbol{x}) \leq 0$ may be violated if it is extremely close to a boundary. Instead, a conservative safety constraints are applied. It is defined as:

$$-f^i(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \geq h, \ \forall i = 1, \ldots, m \tag{2.99}$$

where $h > 0$ is a safety threshold. With the conservative safety constraints, the decision points will keep a distance of h to the actual boundary to increase robustness.

In searching for the acceptable step length, the conservative safety constraints shall be satisfied in addition to the requirement of sufficient decrease.

With above consideration, the safe step length selection algorithm is formulated in Algorithm 4

Algorithm 4 Safe step length selection for exact measurements (e-StepLength)

Input: Search direction $\boldsymbol{p}_k$, gradient estimator for objective $G^0(\boldsymbol{x}_k, \nu_k)$, gradient estimator for constraints $\hat{G}^i(\boldsymbol{x}_k, \nu_k)$, centers and radii $\hat{\mathcal{O}}_k^i$, $\hat{\mathcal{R}}_k^i$, $i = 1, \ldots, m$, $\hat{\alpha}_k > 0$, $h > 0$, $\rho \in (0, 1)$, $c \in (0, 1)$;
while $\min\limits_{i=1,\ldots,m} \left\{ \hat{\mathcal{R}}_k^i - \| (\boldsymbol{x}_k + \hat{\alpha}_k \boldsymbol{p}_k) - \hat{\mathcal{O}}_k^i \|_2 \right\} \leq 0$ **do**
    $\hat{\alpha}_k = \rho \hat{\alpha}_k$;
end while
Set $\alpha_k \leftarrow \hat{\alpha}_k$;
Evaluate $f^i(\boldsymbol{x}_k)$ and $f^i(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)$, $i = 0, \ldots, m$;
while $f^0(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \geq f^0(\boldsymbol{x}_k) + c\alpha_k G^0(\boldsymbol{x}_k, \nu_k)^{\mathrm{T}} \boldsymbol{p}_k$ **or** $\min\limits_{i=1,\ldots,m} \{-f^i(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)\} < h$ **do**
    $\alpha_k = \rho \alpha_k$;
    Update $f^i(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)$, $i = 0, \ldots, m$;
end while
Return: α_k

2.5.7 Safe-line-search algorithm (e-SLS) formulation

With the determined search direction $\boldsymbol{p}_k$, the safe set $\mathcal{S}_k$ and the acceptable step length that provides sufficient decrease in the objective function and satisfies the safety constraints. The safe-line-search algorithm for exact measurement (e-SLS) can be formulated. An outline of the algorithm is summarized as:

1. Compute the gradient estimator $\hat{G}^i(\boldsymbol{x}_k)$ for $f^i(\boldsymbol{x}_k),\ i = 0, \ldots, m$;
2. Determine the search direction $\boldsymbol{p}_k$;
3. Compute the search direction projection $\boldsymbol{p}_k^{proj}$ if $\min\limits_{i=1,\ldots,m} -f^i(x_k)$ is below a safety threshold h;
4. Find the local safe set $\mathcal{S}_k$;
5. Compute a safety-guaranteed step length α_k;
6. Set the next point as $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$;
7. Evaluate functions at $f^i(\boldsymbol{x}_{k+1}),\ i = 0, \ldots, m$;
8. Check the convergence condition.

The complete algorithm is shown in Algorithm 5.

Algorithm 5 Safe-line-search algorithm for exact measurements (e-SLS)

Input: $\boldsymbol{x}_0 \in \mathcal{D}$, $f^i(\boldsymbol{x}_0), i = 0, \ldots, m$, L, $M > 0$, $d > 0$, $\mu > 0$, $h > 0$, $\epsilon > 0$, $\rho \in (0, 1)$, $c \in (0, 1)$;
for $k = 0, 1, \ldots$ **do**
  Take $\nu_k = \min\left\{\frac{2\mu}{\sqrt{d}M}, \frac{\min\limits_{i=1,\ldots,m}\{-f^i(\boldsymbol{x}_k)\}}{2L}\right\}$;
  For each $f^i(\boldsymbol{x})$, $i = 0, \ldots, m$, make evaluations at points $\boldsymbol{x}_k + \nu_k \boldsymbol{e}_j$, $j = 1, \ldots, d$;
  Compute the gradient estimators $G^i(\boldsymbol{x}_k, \nu_k)$ of $\nabla f^i(\boldsymbol{x}_k)$, $i = 0, \ldots, m$ using (2.17);
  Set $\boldsymbol{p}_k = -G^0(\boldsymbol{x}_k, \nu_k)$ **or** *Quasi-Newton* direction (2.42);
  Compute $\hat{G}^i(\boldsymbol{x}_k, \nu_k)$, $i = 1, \ldots, m$ (2.46);
  Compute the centers $\hat{\mathcal{O}}_k^i$ and radii $\hat{\mathcal{R}}_k^i$ of safe set $\hat{\mathcal{S}}_k^i$, $i = 1, \ldots, m$ (2.57), (2.58);
  if $\min\{-f^i(\boldsymbol{x}_k)\} \leq h$ **then**
    $\mathcal{A} := \{i : -f^i(\boldsymbol{x}_k)\} \leq h\}$;
    $\boldsymbol{C}_k = \left[\hat{G}^i(\boldsymbol{x}_k, \nu_k), \ldots, \hat{G}^j(\boldsymbol{x}_k, \nu_k)\right]$, $i, \ldots, j \in \mathcal{A}$;
    $\boldsymbol{\lambda}_{sol} \leftarrow$ Solve Non-negative LS problem 3;
    $\boldsymbol{p}_k^{proj} = \boldsymbol{p}_k - \boldsymbol{C}_k \boldsymbol{\lambda}_{sol}$;
    $\boldsymbol{p}_k = \boldsymbol{p}_k^{proj}$;
  end if
  Find $\alpha_k \leftarrow$ e-StepLength 4;
  Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$;
  Evaluate $f^i(\boldsymbol{x}_{k+1})$, $i = 0, \ldots, m$;
  if $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2 \leq \epsilon$ **then**
    $\boldsymbol{x}_* = \boldsymbol{x}_k$;
    $\boldsymbol{\lambda}_* = \boldsymbol{0}$;
    if $\mathcal{A} \neq \emptyset$ **then**
      $\lambda_*^i = \lambda_{sol}^i$, $i \in \mathcal{A}$;
    end if
    Algorithm Stop;
  end if
end for

2.6 Safe-line-search algorithm for noisy measurements (n-SLS)

2.6.1 Influence of measurement noises

One of the main challenges of model-free optimization methods is the measurement noises. Unlike function evaluations in simulation, in practical implementation, the measurements of function values are always corrupted with noises. Those noises impose two main limitations on the safe derivative-free optimization algorithm:

1. It reduces the accuracy of safety constraints function measurements;
2. It significantly enlarges the estimation error of gradient estimators.

In the following sections, the modification of the algorithm for exact measurements for handling measurement noises are discussed. Firstly, some notations and assumption are made.

Denote by ξ the noise variable, and assume that ξ is independently and identically distributed (i.i.d) zero-mean-σ-sub-Gaussian 3. A sub-Gaussian variable X denoted by $X \sim \text{subG}(\sigma^2)$, has the following property:

Theorem 4. *Let $X \sim subG(\sigma^2)$. Then for any $b > 0$, there is:*

$$\mathbb{P}[X > b] \leq \exp\left(-\frac{b^2}{2\sigma^2}\right) \tag{2.100}$$

$$\mathbb{P}[X < -b] \leq \exp\left(-\frac{b^2}{2\sigma^2}\right) \tag{2.101}$$

If $X_1, \ldots, X_n$ are i.i.d subG(σ^2), then the average $\bar{X} = \frac{1}{n}\sum_{i=1}^{n}$ is:

$$\bar{X} \sim \text{subG}(\frac{\sigma^2}{n}) \tag{2.102}$$

Using (2.100) and (2.101), $\bar{X}$ satisfies:

$$\mathbb{P}[\bar{X} > b] \leq \exp\left(-\frac{nb^2}{2\sigma^2}\right) \tag{2.103}$$

$$\mathbb{P}[\bar{X} < -b] \leq \exp\left(-\frac{nb^2}{2\sigma^2}\right) \tag{2.104}$$

With the assumption on measurement noise as sub-Gaussian random variable, using its properties, the modification of safe-line-search algorithm for noisy measurements is stated below.

2.6.2 Modification of e-SLS algorithm for noisy measurements

The modification for noisy measurements is based on the idea that replacing variables of single measurements by a sample-average ones can reduce the variances of uncertain variables. The modification for individual component in the Algorithm 5 is derived below.

The first modification is on the Gradient estimator. Let n_k be the number of repeated measurements, ξ^i_{0l} be the measurement noise of l-th measurement of constraint $f^i(\boldsymbol{x}_k)$, $\tilde{f}^i(\boldsymbol{x}_k;\xi^i_{0l})$ be the l-th noise-corrupted measurement of $f^i(\boldsymbol{x}_k)$:

$$\tilde{f}^i(\boldsymbol{x}_k;\xi^i_{0l}) = f^i(\boldsymbol{x}_k) + \xi^i_{0l} \tag{2.105}$$

and $\tilde{f}^i(\boldsymbol{x}_k+\nu_k\boldsymbol{e}_j;\xi^i_{jl})$ be the l-th measurement of $f^i(\boldsymbol{x}_k+\nu_k\boldsymbol{e}_j)$

$$\tilde{f}^i(\boldsymbol{x}_k+\nu_k\boldsymbol{e}_j;\xi^i_{jl}) = f^i(\boldsymbol{x}_k+\nu_k\boldsymbol{e}_j) + \xi^i_{jl} \tag{2.106}$$

Let $\bar{f}^i(\boldsymbol{x}_k;\xi^i_0)$ and $\bar{f}^i(\boldsymbol{x}_k+\nu_k\boldsymbol{e}_j;\xi^i_j)$ be the mean values of $\tilde{f}^i(\boldsymbol{x}_k;\xi^i_{0l})$ and $\tilde{f}^i(\boldsymbol{x}_k+\nu_k\boldsymbol{e}_j;\xi^i_{jl})$.

$$\bar{f}^i(\boldsymbol{x}_k;\xi^i_0) = \frac{\sum_{l=1}^{n_k}\tilde{f}^i(\boldsymbol{x}_k;\xi^i_{0l})}{n_k} \tag{2.107}$$

$$\bar{f}^i(\boldsymbol{x}_k+\nu_k\boldsymbol{e}_j;\xi^i_j) = \frac{\sum_{l=1}^{n_k}\tilde{f}^i(\boldsymbol{x}_k+\nu_k\boldsymbol{e}_j;\xi^i_{jl})}{n_k} \tag{2.108}$$

A sample-average gradient estimator $\bar{G}^i(\boldsymbol{x}_k,\nu_k;\xi^i)$ of $\nabla f^i(\boldsymbol{x}_k)$ is computed by:

$$\bar{G}^i(\boldsymbol{x}_k,\nu_k;\xi^i) = \sum_{j=1}^{d}\frac{\bar{f}^i(\boldsymbol{x}_k+\nu_k\boldsymbol{e}_j;\xi^i_j) - \bar{f}^i(\boldsymbol{x}_k;\xi^i_0)}{\nu_k}\boldsymbol{e}_j \tag{2.109}$$

In case of noisy measurement, the estimation error becomes larger than the case of exact measurements, and it comes from both the finite difference approximation, and the measurement noises. According to the *Law of large numbers*, the estimation error from noisy measurements can be reduced by increasing the number of measurements n_k, since with larger n_k, the average tends to be closer to the expected value.

Denote the gradient estimation error with noisy measurements by Δ^i_{k,ξ^i}:

$$\Delta^i_{k,\xi^i} = \bar{G}^i(\boldsymbol{x}_k,\nu_k;\xi^i) - \nabla f^i(\boldsymbol{x}_k) \tag{2.110}$$

The upper bound of Δ^i_{k,ξ^i} can be determined by the following.

Theorem 5. *The gradient estimation error $\|\Delta^i_{k,\xi^i}\|_2$ with zero-mean-σ-sub-Gaussian noise is upper bounded with high probability $1-\delta$ by:*

$$\mathbb{P}\left\{\|\Delta^i_{k,\xi^i}\|_2 \le \sqrt{\frac{dM^2\nu_k^2}{4} - \frac{4d\sigma^2\ln\delta}{\nu_k^2 n_k}}\right\} \ge 1-\delta \tag{2.111}$$

To balance the two terms inside the square root, let $n_k = -\frac{16\sigma^2 \ln \delta}{3\nu_k^4 M^2}$ *, then*

$$\mathbb{P}\left\{\|\Delta^i_{k,\xi^i}\|_2 \leq \sqrt{d} M \nu_k\right\} \geq 1 - \delta \tag{2.112}$$

Proof. *Let* ξ^i_j *be the mean value of* ξ^i_{jl}*:*

$$\xi^i_j = \sum_{l=1}^{n_k} \xi^i_{jl},\ j = 0, \ldots, d \tag{2.113}$$

Recall the M-smooth property 2 of $f^i(\boldsymbol{x}_k)$*:*

$$f^i(\boldsymbol{x}_k + \nu_k \boldsymbol{e}_j) \leq f^i(\boldsymbol{x}_k) + \nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}(\nu_k \boldsymbol{e}_j) + \frac{1}{2} M \|\nu_k \boldsymbol{e}_j\|_2^2 \tag{2.114}$$

Using (2.109) and (2.110), the estimation error can be written as.

$$\|\Delta^i_{k,\xi^i}\|_2 = \sqrt{\sum_{j=1}^{d} \left[\frac{f^i(\boldsymbol{x}_k + \nu_k \boldsymbol{e}_j) - f^i(\boldsymbol{x}_k) + \xi^i_j - \xi^i_0}{\nu_k} - \nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}} \boldsymbol{e}_j\right]^2 \|\boldsymbol{e}_j\|_2^2} \tag{2.115}$$

$$\leq \sqrt{\sum_{j=1}^{d} \left[\frac{f^i(\boldsymbol{x}_k + \nu_k \boldsymbol{e}_j) - f^i(\boldsymbol{x}_k) - \nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}(\nu_k \boldsymbol{e}_j)}{\nu_k}\right]^2 + \frac{1}{\nu_k^2} \sum_{j=1}^{d} \left(\xi^i_j - \xi^i_0\right)^2} \tag{2.116}$$

$$\leq \sqrt{\sum_{j=1}^{d} \left[\frac{M\|\nu_k \boldsymbol{e}_j\|_2^2}{2\nu_k}\right]^2 + \frac{1}{\nu_k^2} \sum_{j=1}^{d} \left(\xi^i_j - \xi^i_0\right)^2} \tag{2.117}$$

Let $\tilde{\xi}^i = \xi^i_j - \xi^i_0$*, since* $\xi^i_{jl} \sim \mathrm{subG}(\sigma^2)$*,* $j = 0, \ldots, d$*, according to (2.102), there is:*

$$\xi^i_j \sim \mathrm{subG}(\frac{\sigma^2}{n_k}),\ j = 0, \ldots, d \tag{2.118}$$

and

$$\tilde{\xi}^i \sim \mathrm{subG}(\frac{2\sigma^2}{n_k}) \tag{2.119}$$

then using 2.103 there is:

$$\mathbb{P}\left\{\bar{\xi}^i > b\right\} \leq \exp\left(-\frac{n_k b^2}{4\sigma^2}\right) = \delta \tag{2.120}$$

$$b^2 = -\frac{4\sigma^2 \ln \delta}{n_k} \tag{2.121}$$

thus:

$$\mathbb{P}\left\{\bar{\xi^i}^2 < -\frac{4\sigma^2 \ln \delta}{n_k}\right\} \geq 1-\delta \tag{2.122}$$

then (2.117) is equivalent to:

$$\mathbb{P}\left\{\|\Delta^i_{k,\xi^i}\|_2 \leq \sqrt{\frac{dM^2\nu_k^2}{4} - \frac{4d\sigma^2 \ln \delta}{\nu_k^2 n_k}}\right\} \geq 1-\delta \tag{2.123}$$

When $n_k = -\frac{16\sigma^2 \ln \delta}{3\nu_k^4 M^2}$*, there is:*

$$\mathbb{P}\left\{\|\Delta^i_{k,\xi^i}\|_2 \leq \sqrt{d}M\nu_k\right\} \geq 1-\delta \tag{2.124}$$

□

Thus, in case of noisy measurements, the gradient estimator $G^i(\boldsymbol{x}_k, \nu_k)$ is replaced by $\bar{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)$ (2.109) and the estimation deviation becomes Δ^i_{k,ξ^i} (2.111).

The next component to be modified is the safe set $\mathcal{S}_k$ at each iteration. With the upper bound of the estimation deviation $\|\Delta^i_{k,\xi^i}\|_2$, a conservative gradient estimator can be made by the following:

Theorem 6. *Let* $\boldsymbol{p}_k$ *be the descent direction with unit length,* $G^i(\boldsymbol{x}_k, \nu_k; \xi^i)$ *be the gradient estimator from noisy measurements,* $\|\Delta^i_{k,\xi^i}\|_2$ *be the estimation error and* $n_k = -\frac{16\sigma^2 \ln \delta}{3\nu_k^4 M^2}$*, then with high probability* $1-\delta$*, all points* $\boldsymbol{x}_{k+1}$ *satisfying the following constraints are feasible:*

$$\bar{f}^i(\boldsymbol{x}_k; \xi_0^i) + \hat{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)^{\mathrm{T}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \frac{1}{2}M\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \leq 0 \tag{2.125}$$

where

$$\hat{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i) = \bar{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i) + \sqrt{\frac{dM^2\nu_k^2}{4} - \frac{4d\sigma^2 \ln \delta}{\nu_k^2 n_k}}\frac{\|\boldsymbol{p}_k\|_2 \boldsymbol{e}}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_k} \tag{2.126}$$

with $n_k = -\frac{16\sigma^2 \ln \delta}{3\nu_k^4 M^2}$*, then:*

$$\hat{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i) = \bar{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i) + \frac{\sqrt{d}M\nu_k\|\boldsymbol{p}_k\|_2}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_k}\boldsymbol{e} \tag{2.127}$$

Proof. *For simplicity in this proof, let* $n_k = -\frac{16\sigma^2 \ln \delta}{3\nu_k^4 M^2}$*. Similar to Proof 2.5.4, if* $\boldsymbol{x}_{k+1}$ *is feasible, there is*

$$\bar{f}^i(\boldsymbol{x}_k; \xi_0^i) + \nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \frac{1}{2}M\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \leq 0 \tag{2.128}$$

Given (2.125), proving (2.128) is equivalent to prove the following (as shown in Proof 2.5.4):

$$\|\nabla f^i(\boldsymbol{x}_k)\|_2 \cos\alpha - \|\bar{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)\|_2 \cos\beta \leq \sqrt{d} M \nu_k \tag{2.129}$$

where α is the angle between gradient $\nabla f^i(\boldsymbol{x}_k)$ and search direction $\boldsymbol{p}_k$, β is the angle between $\bar{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)$ and $\boldsymbol{p}_k$. Let ψ be the angle between $\bar{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i) - \nabla f^i(\boldsymbol{x}_k)$ and $-\boldsymbol{p}_k$. Similar to Proof 2.5.4, there is

$$\left(\|\nabla f^i(\boldsymbol{x}_k)\|_2 \cos\alpha - \|\bar{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)\|_2 \cos\beta\right)^2 = (\|\Delta^i_{k^i,\xi^i}\|_2 \cos\psi)^2 \tag{2.130}$$

$$\|\nabla f^i(\boldsymbol{x}_k)\|_2 \cos\alpha - \|\bar{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)\|_2 \cos\beta \leq \|\Delta^i_{k^i,\xi^i}\|_2 \tag{2.131}$$

Since with $n_k = -\frac{16\sigma^2 \ln\delta}{3\nu_k^4 M^2}$, (2.112) holds, then

$$\mathbb{P}\left\{\|\nabla f^i(\boldsymbol{x}_k)\|_2 \cos\alpha - \|\bar{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)\|_2 \cos\beta \leq \sqrt{d} M \nu_k\right\} \geq 1 - \delta \tag{2.132}$$

thus

$$\mathbb{P}\left\{\bar{f}^i(\boldsymbol{x}_k; \xi^i_0) + \hat{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)^{\mathrm{T}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \frac{1}{2} M \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \leq 0\right\} \geq 1 - \delta \tag{2.133}$$

□

Then, the safe set for $f^i(\boldsymbol{x}_k)$ at $\boldsymbol{x}_k$ with noisy measurements is modified as:

$$\hat{\mathcal{S}}^i_{k,\xi^i} := \left\{\boldsymbol{x} \in \mathbb{R}^d : \bar{f}^i(\boldsymbol{x}_k) + \hat{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2} M \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 \leq 0\right\} \tag{2.134}$$

This safe set is a ball centered at $\hat{\mathcal{O}}^i_{k,\xi^i}$ with radius $\hat{\mathcal{R}}^i_{k,\xi^i}$:

$$\hat{\mathcal{O}}^i_{k,\xi^i} = \boldsymbol{x}_k - \frac{1}{M} \hat{G}(\boldsymbol{x}_k, \nu_k; \xi^i) \tag{2.135}$$

$$\hat{\mathcal{R}}^i_{k,\xi^i} = \sqrt{\frac{1}{M}\left(\frac{\|\hat{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)\|_2^2}{M} - 2\bar{f}^i(\boldsymbol{x}_k; \xi^i_0)\right)} \tag{2.136}$$

The overall safe set $\mathcal{S}_{k,\xi}$ for all constraints $f^i(\boldsymbol{x})$, $i = 1, \ldots, m$ is the intersection of $\hat{\mathcal{S}}^i_{k,\xi^i}$

$$\mathcal{S}_{k,\xi} := \left\{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x} - \hat{\mathcal{O}}^i_{k,\xi^i}\|_2 \leq \hat{\mathcal{R}}^i_{k,\xi^i}, \forall = 1, \ldots, m\right\} \tag{2.137}$$

The search direction projection has the same procedure in section sec:direction projection for the case of exact measurements, except that the $\boldsymbol{C}_k$ matrix consists

of $\hat{G}^i(\boldsymbol{x}_k, \nu_k; \xi^0), i \in \mathcal{A}$ and $\boldsymbol{p}_k$ is $\bar{G}^0(\boldsymbol{x}_k, \nu_k; \xi^0)$.

In addition to the above modification, the step length selection method shall be adapted accordingly. The uncertain gradient estimators and function measurements are replaced by sample-average counterparts.

The step length upper bound in the Algorithm 4 is changed to the following:

$$\hat{\alpha}_{k,\xi} = \max\{\alpha \in \mathbb{R} : \|\boldsymbol{x}_k + \alpha \boldsymbol{p}_k - \hat{\mathcal{O}}^i_{k,\xi^i}\|_2 < \hat{\mathcal{R}}^i_{k,\xi^i}, \forall i = 1, \ldots, m, \alpha > 0\} \quad (2.138)$$

and the modified step length selection algorithm for noisy measurements is shown below:

Algorithm 6 Safe step length selection for noisy measurements (n-StepLength)

Input: Search direction $\boldsymbol{p}_k$, gradient estimator for objective $\bar{G}^0(\boldsymbol{x}_k, \nu_k; \xi^0)$, gradient estimator for constraints $\hat{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)$, centers and radii $\hat{\mathcal{O}}^i_{k,\xi^i}$, $\hat{\mathcal{R}}^i_{k,\xi^i}$, $i = 1, \ldots, m$, $\hat{\alpha}_{k,\xi} > 0$, $h > 0$, $n_k > 0$, $\rho \in (0, 1)$, $c \in (0, 1)$;
while $\min\limits_{i=1,\ldots,m} \left\{\hat{\mathcal{R}}^i_{k,\xi^i} - \| (\boldsymbol{x}_k + \hat{\alpha}_{k,\xi}\boldsymbol{p}_k) \|_2 - \hat{\mathcal{O}}^i_{k,\xi^i}\right\} \leq 0$ **do**
    $\hat{\alpha}_{k,\xi} = \rho\hat{\alpha}_{k,\xi}$;
end while
Set $\alpha_{k,\xi} \leftarrow \hat{\alpha}_{k,\xi}$;
Compute $\bar{f}^i(\boldsymbol{x}_k; \xi^i_0)$ and $\bar{f}^i(\boldsymbol{x}_k + \alpha_{k,\xi}\boldsymbol{p}_k; \xi^i_0),\ i = 0, \ldots, m$ using (2.107)
while $\bar{f}^0(\boldsymbol{x}_k + \alpha_{k,\xi}\boldsymbol{p}_k; \xi^0_0) \geq \bar{f}^0(\boldsymbol{x}_k; \xi^0_0) + c\alpha_{k,\xi}\bar{G}^0(\boldsymbol{x}_k, \nu_k; \xi^0)^{\mathrm{T}}\boldsymbol{p}_k$ **or** $\min\limits_{i=1,\ldots,m} \left\{-\bar{f}^i(\boldsymbol{x}_k + \alpha_{k,\xi}\boldsymbol{p}_k; \xi^i_0))\right\} < h$ **do**
    $\alpha_{k,\xi} = \rho\alpha_{k,\xi}$;
    Update $\bar{f}^i(\boldsymbol{x}_k + \alpha_{k,\xi}\boldsymbol{p}_k; \xi^i_0),\ i = 0, \ldots, m$;
end while
Return: $\alpha_{k,\xi}$;

2.6.3 Safe-line-search algorithm (n-SLS) formulation

With the above modifications, the safe-line-search algorithm for noisy measurements is formulated below.

Algorithm 7 Safe-line-search algorithm for noisy measurements (n-SLS)

Input: $\boldsymbol{x}_0 \in \mathcal{D}$, $f^i(\boldsymbol{x}_0), i = 0, \ldots, m$, L, $M > 0$, $d > 0$, $\mu > 0$, $h > 0$, $\epsilon > 0$, $\sigma > 0$, $\rho \in (0, 1)$, $c \in (0, 1)$, $\delta \in [0, 1]$;
for $k = 0, 1, \ldots$ **do**
  Take $\nu_k = \min\left\{\frac{2\mu}{\sqrt{d}M}, \frac{\min\limits_{i=1,\ldots,m}\{-\bar{f}^i(\boldsymbol{x}_k;\xi_0^i)\}}{2L}\right\}$, $n_k = -\frac{16\sigma^2 \ln\delta}{3\nu_k^4 M^2}$;
  For each $f^i(\boldsymbol{x})$, $i = 0, \ldots, m$, make evaluations at points $\boldsymbol{x}_k + \nu_k e_j$, $j = 1, \ldots, d$, measure each value by n_k times;
  Compute the gradient estimators $\bar{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)$ of $\nabla f^i(\boldsymbol{x}_k)$, $i = 0, \ldots, m$ using (2.109);
  Set $\boldsymbol{p}_{k,\xi} = -\bar{G}^0(\boldsymbol{x}_k, \nu_k; \xi^0)$ **or** *Quasi-Newton* direction (2.42);
  Compute $\hat{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i)$, $i = 1, \ldots, m$ (2.126);
  Compute the centers $\hat{\mathcal{O}}^i_{k,\xi^i}$ and radii $\hat{\mathcal{R}}^i_{k,\xi^i}$ of safe set $\hat{\mathcal{S}}^i_{k,\xi^i}$, $i = 1, \ldots, m$ (2.135), (2.136);
  if $\min\{-\bar{f}^i(\boldsymbol{x}_k; \xi_0^i)\} \le h$ **then**
    $\mathcal{A} := \left\{i : -\bar{f}^i(\boldsymbol{x}_k; \xi_0^i)\} \le h\right\}$;
    $\boldsymbol{C}_k = \left[\hat{G}^i(\boldsymbol{x}_k, \nu_k; \xi^i), \ldots, \hat{G}^j(\boldsymbol{x}_k, \nu_k; \xi^i)\right]$, $i, \ldots, j \in \mathcal{A}$;
    $\boldsymbol{\lambda}_{sol} \leftarrow$ Solve Non-negative LS problem 3;
    $\boldsymbol{p}^{proj}_{k,\xi} = \boldsymbol{p}_{k,\xi} - \boldsymbol{C}_k \boldsymbol{\lambda}_{sol}$;
    $\boldsymbol{p}_{k,\xi} = \boldsymbol{p}^{proj}_{k,\xi}$;
  end if
  Find $\alpha_{k,\xi} \leftarrow$ n-StepLength 6;
  Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_{k,\xi}\boldsymbol{p}_{k,\xi}$;
  Evaluate $f^i(\boldsymbol{x}_{k+1})$, $i = 0, \ldots, m$, measure each value by n_k times;
  if $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k \le \epsilon$ **then**
    $\boldsymbol{x}_* = \boldsymbol{x}_k$;
    $\boldsymbol{\lambda}_* = \boldsymbol{0}$;
    if $\mathcal{A} \neq \emptyset$ **then**
      $\lambda^i_* = \lambda^i_{sol}$, $i \in \mathcal{A}$;
    end if
    Algorithm Stop;
  end if
end for

2.7 Convergence and safety analysis

2.7.1 Convergence analysis

Theorem 7. *With sufficiently large iterations, the e-SLS converges to a KKT point.*

Proof. *If the optimal point $\boldsymbol{x}_*$ is inside of the feasible set $\mathcal{D}$, recall that iteration has the form*

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \tag{2.139}$$

where α_k is selected by Algorithm 4, and $\boldsymbol{p}_k$ is a descend direction.

Let ψ_k be the angle between $\boldsymbol{p}_k$ and negative gradient $-G^0(\boldsymbol{x}_k, \nu_k)$, then as k increases, there is

$$\sum_{k>0} \cos^2 \psi_k \|G^0(\boldsymbol{x}_k, \nu_k)\|^2 < \infty \tag{2.140}$$

If $\boldsymbol{p}_k$ is the steepest descend direction, then $\cos \psi_k = 1$, $k > 0$, therefore

$$\sum_{k>0} \|G^0(\boldsymbol{x}_k, \nu_k)\|^2 < \infty \tag{2.141}$$

which implies that

$$\lim_{k\to\infty} \|G^0(\boldsymbol{x}_k, \nu_k)\| = 0 \tag{2.142}$$

Thus, the point $\boldsymbol{x}_k = \arg\left(\lim_{k\to\infty} \|G^i(\boldsymbol{x}_k, \nu_k)\|\right)$, and the Lagrangian multipliers $\boldsymbol{\lambda}_k = \mathbf{0}$, is a KKT point $[\boldsymbol{x}_k, \mathbf{0}]$.

If the optimal point is outside of feasible set $\mathcal{D}$, let $\mathcal{A}$ be the set of indices of approximately active constraints at the optimal feasible point $\boldsymbol{x}_*$, that is:

$$\mathcal{A} := \left\{ i : \ -f^i(\boldsymbol{x}_*) \le h \right\} \tag{2.143}$$

where h is a safety threshold. From line 9 of Algorithm 5, the descent direction $\boldsymbol{p}_k$ is projected on $\boldsymbol{p}_k^{proj}$. From Proof 2.5.5, it is known that

$$\measuredangle(\boldsymbol{p}_k^{proj}, \boldsymbol{p}_k) \le \frac{\pi}{2} \tag{2.144}$$

and

$$\frac{\pi}{2} \le \measuredangle(\boldsymbol{p}_k^{proj}, G^i(\boldsymbol{x}_k, \nu_k)) \le \frac{3\pi}{2}, \ \forall i \in \mathcal{A} \tag{2.145}$$

That is, the direction $\boldsymbol{p}_k^{proj}$ is a descent direction without increasing active constraints. From line 18 of the algorithm, if the convergence condition is satisfied at point $\boldsymbol{x}_k$, then $\alpha_k \boldsymbol{p}_k^{proj} \to 0$, thus either $\alpha \to 0$ or $\|\boldsymbol{p}_k^{proj}\|_2 \to 0$.

If $\alpha \to 0$, then from the Armijo condition (2.95), there is:

$$f^0(\boldsymbol{x}_k + 0_+ \boldsymbol{p}_k^{proj}) \ge f^0(\boldsymbol{x}_k) + 0_+ c \nabla f^0(\boldsymbol{x}_k)^{\mathrm{T}} \boldsymbol{p}_k^{proj} \tag{2.146}$$

$$f^0(\boldsymbol{x}_{k-1}) \geq f^0(\boldsymbol{x}_k) \tag{2.147}$$

that is, all points around $\boldsymbol{x}_k + 0_+\boldsymbol{p}_k^{proj}$ *have a larger value than* $f^0(\boldsymbol{x}_k)$*, and*

$$\nabla f^0(\boldsymbol{x}_{k-1})^{\mathrm{T}}\boldsymbol{p}_{k-1}^{proj} \leq 0 \tag{2.148}$$

$$\nabla f^0(\boldsymbol{x}_k + 0_+\boldsymbol{p}_k^{proj})^{\mathrm{T}}\boldsymbol{p}_k^{proj} \geq 0 \tag{2.149}$$

which implies that $\|\boldsymbol{p}_k^{proj}\|_2 \to 0$*.*

Thus, from $\alpha_k\boldsymbol{p}_k^{proj} \to 0$*, there is* $\|\boldsymbol{p}_k^{proj}\|_2 \to 0$*. Then, from line 13 of the algorithm, there is* $\boldsymbol{p}_k \approx \boldsymbol{C}_k\boldsymbol{\lambda}_{sol}$*, which means that the gradient* $-G^0(\boldsymbol{x}, \nu_k)$ *can be linearly expressed by* $G^i(\boldsymbol{x}_k, \nu_k),\ i \in \mathcal{A}$*. From Lemma 4 we know that* $[\boldsymbol{x}_k, \boldsymbol{\lambda}_{sol}, \boldsymbol{0}]$ *is a KKT point.* □

2.7.2 Safety analysis

Theorem 8. *With gradient estimator* $\hat{G}^i(\boldsymbol{x}_k, \nu_k) = G^i(\boldsymbol{x}_k, \nu_k) + \frac{\sqrt{d}M\nu_k\|\boldsymbol{p}_k\|_2}{2\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_k}\boldsymbol{e}$*, and safety threshold* $h \geq 0$*, e-SLS is safety-guarantee during optimization.*

Proof. *Assume that the unconstrained optimal point* $\boldsymbol{x}_*^{unc}$ *of the objective function* $f^0(\boldsymbol{x})$ *is outside of the feasible set* $\mathcal{D}$*,* $f^i(\boldsymbol{x}),\ i \in \mathcal{A}$ *are the active constraints at the optimal feasible solution* $\boldsymbol{x}_*$*. By Algorithm 4, there is:*

$$\|\boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k - \hat{\mathcal{O}}_k^i\|_2 < \hat{\mathcal{R}}_k^i, \forall i = 1, \ldots, m \tag{2.150}$$

where $\hat{\mathcal{O}}_k^i$ *and* $\hat{\mathcal{R}}_k^i$ *are the center and radius of safe sets* $\hat{\mathcal{S}}_k^i$ *computed with* $\hat{G}^i(\boldsymbol{x}_k, \nu_k) = G^i(\boldsymbol{x}_k, \nu_k) + \frac{\sqrt{d}M\nu_k\|\boldsymbol{p}_k\|_2}{2\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_k}\boldsymbol{e}$*.*

If $\boldsymbol{x}_k$ *is not close to a boundary, that is* $-f^i(\boldsymbol{x}_k) > h,\ \forall i = 1, \ldots, m$*, from Theorem 3, there is:*

$$f^i(\boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k) \leq f^i(\boldsymbol{x}_k) + \alpha_k\nabla f^i(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{p}_k + \frac{1}{2}M\alpha_k^2\|\boldsymbol{p}_k^2\| \leq 0 \tag{2.151}$$

that is, when $\boldsymbol{x}_k$ *is not close to a boundary, all* $\boldsymbol{x}_k$ *are safe.*

If there are some $-f^i(\boldsymbol{x}_k) \leq h$*, then from line 5 in Algorithm 4, the selected step length* α_k *will be such that:*

$$f^i(\boldsymbol{x} + \alpha_k\boldsymbol{p}_k) \leq -h < 0 \tag{2.152}$$

Thus, it can be guaranteed that $\max\{f^i(\boldsymbol{x})\} \leq -h < 0,\ \forall k$*. that is, e-SLS is safe during the optimization process.* □

2.8 Simulation analysis with case studies

2.8.1 Problem description

In this section, the safe line search algorithm for exact measurements and noisy measurements are tested on simulations, and a comparison to other algorithms including the 0-LBM method [UKK19], the DFO-TR method and the SafeOpt-MC method [BKS21] is conducted.

In the simulation, both objective and constraint functions are unknown to the algorithms, while they can only access the function measurements. In addition, the safety requirement that during the optimization process, no point that violates the safety constraints is evaluated shall be fulfilled. Hence it is a model-free safe optimization problem.

The simulation aims to minimize a $2d$ function with different constraints. There are 2 cases in the simulation. In the first case, the constraints are box constraints on the decision variables. In the second case the constraint is changed to a general nonlinear constraint. Both cases are test with exact measurements and noisy measurement respectively.

As this simulation test mainly focus on comparison of the convergence rate of different algorithms, thus a convex objective function is used so that only one global optimum exists. However, the objective function can be a non-convex function. In this case, the algorithm only finds a local minimum and the found minimum point depends on the starting point.

2.8.2 Case 1 Optimization problem with box constraints

In the first case, the algorithm is test on the following optimization problem:

$$\min_{x\in\mathbb{R}^2} \quad (x_1 - 2.7)^2 + 0.5(x_2 - 0.5)^2 - 5 \tag{2.153}$$

$$\text{subject to} \quad x_1 \leq 2.7 \tag{2.154}$$

$$x_2 \geq -5 \tag{2.155}$$

The objective is a convex function with optimal point located at $[2.7, 0.5]$ with optimal value -5. The constraints are box constraints on decision variables.

Firstly, the e-SLS algorithm is tested for solving this problem. The optimization trajectory of e-SLS method is shown in Figure 2.6:

In Figure 2.6, the blue and cyan circles represent the safe set at each iteration for constraints $x_1 \leq 2.7$ and $x_2 \geq -5$. The safe set for one constraint dose not necessarily guarantee safety of other constraints, however, the decision point at each iteration is chosen in the intersection of all safe sets for every constraints, thus the decision points during the optimization process are safe. Besides, one

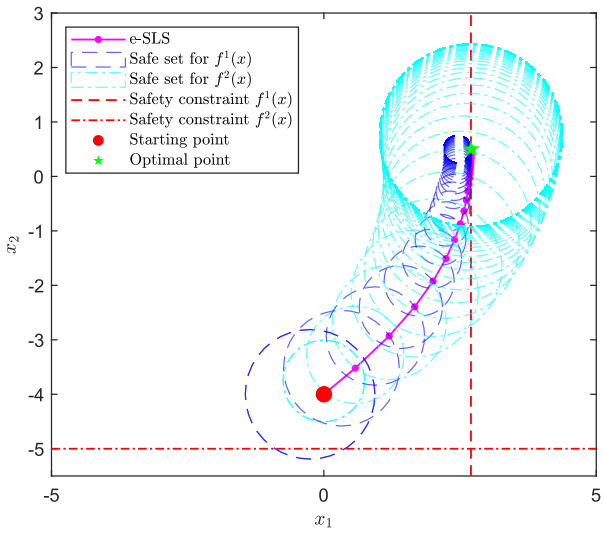


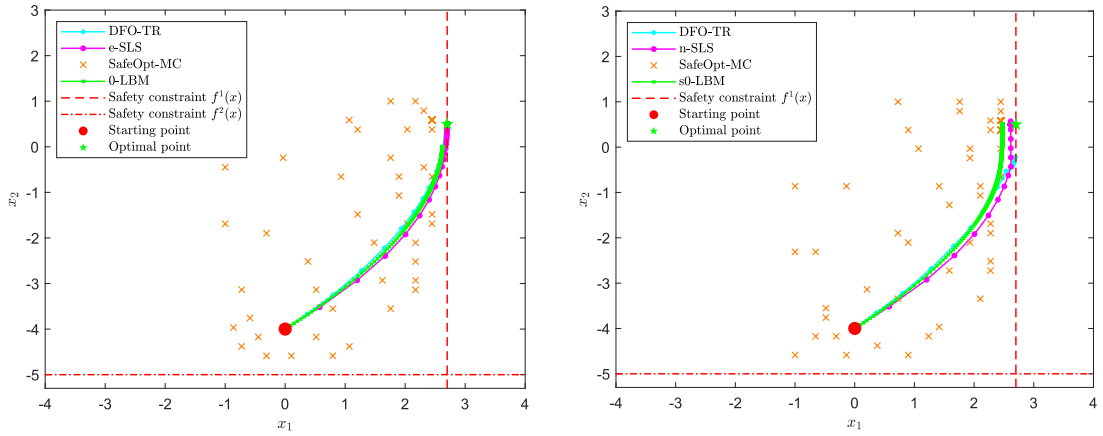
Figure 2.6: Optimization trajectory of e-SLS method in case 1. The red point denotes the starting point, the green star denotes the optimal point, the magenta line is the optimization trajectory of e-SLS method. The red dash lines are the safety constraints, blue and cyan dash circles denote the safe sets at each iteration for safety constraints $f^1(\boldsymbol{x}) \leq 0$ and $f^2(\boldsymbol{x}) \leq 0$ respectively.

can see that the safe set shrinks as the decision point approaching to the corresponding boundary since the radius of safe set depends on the constraint value $f^i(\boldsymbol{x})$. As it converges to 0, the radius reduces. Because every decision points are chosen inside the common parts of all safe sets, throughout the optimization process, the safety constraints are not violated.

Next, a comparison of different algorithms is taken. This comparison includes the DFO-TR method, the e-SLS method, the SafeOpt-MC method and the 0-LBM method. Figure 2.7 shows the optimization trajectories of different algorithms. Figure 2.8 shows the convergence of objective function in different algorithms and Figure 2.9 illustrates the constraint function values.

In Figure 2.7a one can see that the DFO-TR method, e-SLS method perform similar optimization trajectories while the 0-LBM method has a similar trajectory trend to the preceding two methods but with a farther distance to the optimal point when it converges. The SafeOpt-MC method samples a set of safe points to explore the underlying objective and constraint functions. And it repeats sampling the points that give the minimum values in the objective function, which is the points near to the optimal point, after learning the underlying functions. It can be noticed that all methods take measurements within the underlying safe region.

From Figure 2.7b it can seen that, the n-SLS method and s0-LBM method perform a more conservative optimization trajectories compared to the results with exact measurements. They keep a larger distance to the boundary to re-



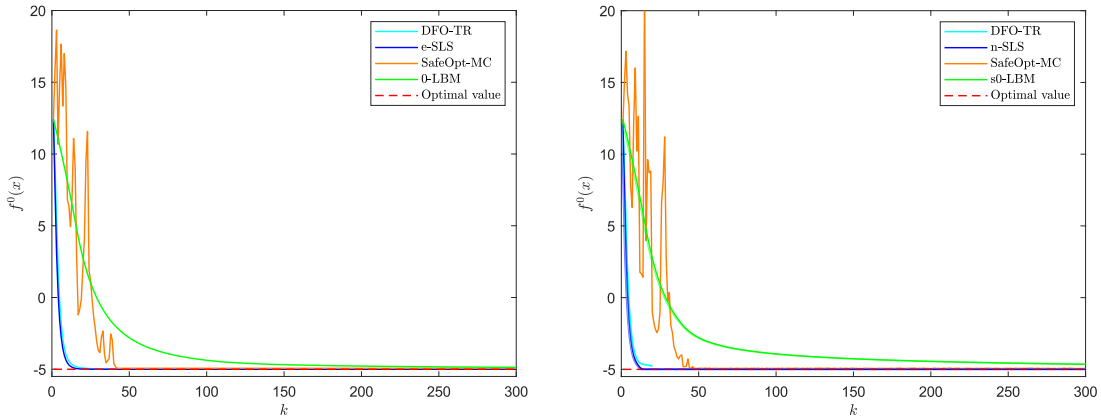
(a) Optimization trajectories of case 1 with exact measurements

(b) Optimization trajectories of case 1 with noisy measurements

Figure 2.7: (a) Optimization trajectory of different algorithms with exact measurements. The red point denotes the starting point, green star denotes the optimal point, red dash lines are the safety constraints. Cyan, magenta, and green lines are the optimization trajectories of DFO-TR method, e-SLS method and 0-LBM method respectively. The orange crosses denote the sample points of SafeOpt-MC method. (b) Optimization trajectories of different algorithms with measurement noise $\xi \sim \mathcal{N}(0, 0.01^2)$.

duce the probability that it violates the safety constraints. The SafeOpt-MC has an intrinsic advantage of handling measurement noises, because it uses Gaussian process to explore and learn the underlying functions, and it considers the influence of measurement noises on the model learning by adding additive adjustment to the covariance matrix of the Gaussian process. Thus, it increases the variance of value of each measurement. The DFO-TR method is influenced by the measurement noises, and it stops updating when approaching to a boundary. This is because that the trust region is computed using estimated gradients computed by finite difference method, which is sensitive to measurement noises. With incorrect trust regions, the optimization problem may be infeasible to the problem solvers.

From Figure 2.8a it can be seen that the e-SLS and DFO-TR methods have similar convergence performance: they both converge to the optimal point within a small number of iterations, and the optimality gap to the true optimal value is small. These two methods have the fastest convergence rate among all methods. The 0-LBM method requires much larger number of iterations to converge to the optimal point, since the step length of 0-LBM method exponentially decreases as it approaches to a boundary. Thus, the closer it is to a boundary, the slower it converges. The SafeOpt-MC method does not provide a monotonous decrease in the objective function, since it first explores the decision space by choosing the



(a) Convergence of objective function of case 1 with exact measurements

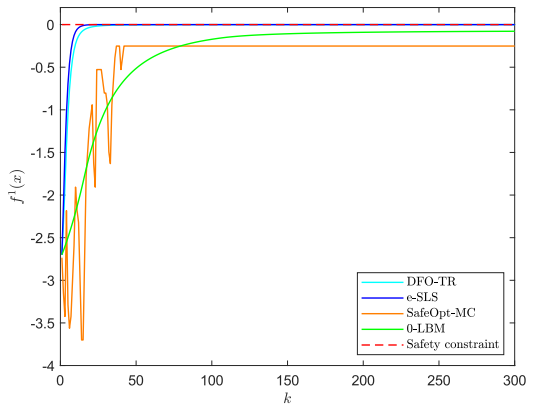
(b) Convergence of objective function of case 1 with noisy measurements

Figure 2.8: (a) Convergence of objective function with exact measurements. Cyan, blue, orange and green lines are the convergence result of DFO-TR method, e-SLS method, SafeOpt-MC method and 0-LBM method respectively. The red dash line is the ground true optimal value. (b) Convergence of objective function with measurement noise $\xi \sim \mathcal{N}(0, 0.01^2)$.

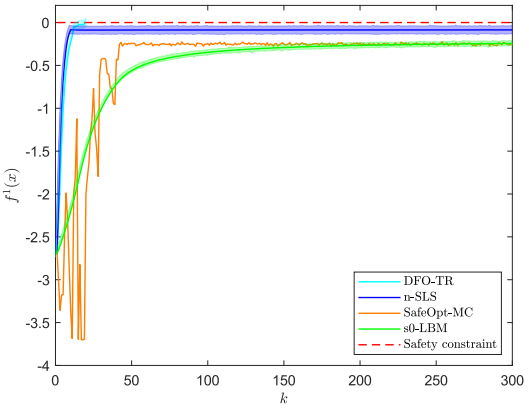
points with the highest uncertainty, then it exploits the function measurements by choosing points that minimize the objective function. During the exploration phase, the algorithm possibly samples points that give higher values in the objective function. Besides, the SafeOpt-MC method discretizes the decision space, therefore, the optimization performance and the computation time much depend on the dimensions of decision space and the density of discretization in each dimension. In this simulation, the search range of SafeOpt-MC method is $[-1, 3]$ in x_1 and $[-5, 1]$ in x_2, and the discretization is taken with 30 points in each dimension. Figure 2.8b shows that the n-SLS, SafeOpt-MC, the s0-LBM methods still perform a good convergence result in the presence of measurement noises, while the DFO-TR method stop converging at early iteration as it contact a boundary. Thus the convergence becomes degraded given the measurement noises.

From Figure 2.9a one can see that all methods are safety guaranteed. The DFO-TR and e-SLS methods approach to the boundary faster than other two methods and in convergence, they are both closer to the boundary as well. The SafeOpt-MC method has the largest distance to the boundary and the 0-LBM method converges lastly. Figure 2.9b shows that the DFO-TR method has some measurements slightly beyond the constraint, while all other methods keep within the safe region.

In addition to the measure of number of iterations, the computation time is another key factor that evaluates an algorithm. As the number of iterations can not present the computation burden of algorithms. Table 2.8.2 shows the



(a) Constraint function of case 1 with exact measurements



(b) Constraint function of case 1 with noisy measurements

Figure 2.9: (a) Constraint function values with exact measurements. Cyan, blue, orange and green lines are the convergence result of DFO-TR method, e-SLS method, SafeOpt-MC method and 0-LBM method respectively. The red dash line is the constraint upper bound. (b) Constraint function values with measurement noise $\xi \sim \mathcal{N}(0, 0.01^2)$.

number of iterations, computation time to convergence and the optimiality gap to the true optimal value of different algorithms with exact measurements.

Table 2.1: Optimization performance of different algorithms

Algorithms	Number of iterations to convergence	Computation time to convergence	Optimality Gap
DFO-TR	22	28.071s	0.240%
e-SLS	**19**	0.060s	**0.083%**
SafeOpt-MC	42	35.872	1.342%
0-LBM	300	**0.031s**	2.601%

From Table 2.8.2 one can see that the SafeOpt-MC method has the least number of iterations, while the 0-LBM has the largest. The e-SLS has a comparable result to the SafeOpt-MC method. For the computation time, however, the SafeOpt-MC requires the longest computation time, since updating the Gaussian process model is time consuming, and it is exponentially growth with the number of dimensions of the problem. The 0-LBM has the fastest computation time. For the optimality gap, the e-SLS method has the lowest deviation to the true optimal value among all methods. From this table, it can be concluded that the e-SLS method performs a good trade-off between the optimization performance and the computation time.

Lastly, to study the effect of starting point near to a boundary on the conver-

gence performance of different algorithms, one complementary simulation is run with an starting point at $\boldsymbol{x}_{start} = [0, -4.99]$, which is further close to a boundary. The simulation results is shown in Figure 2.10.

By this example, one can see that the convergence performances of DFO-TR, e-SLS and SafeOpt-MC methods are not much affected by the changing of starting point to the vicinity of a boundary, while the convergence of 0-LBM method suffer a large lag in the beginning of the optimization. The reason for the lag is that the step length of 0-LBM method is small when it is close to a boundary, regardless the search direction toward deviating the boundary. The DFO-TR and e-SLS methods build a safe region around current point, and safe region has larger space on the side away from the boundary than the side towards the boundary. Thus, if the search direction deviates from the boundary, then the algorithm is not restricted by the small distance to the boundary and still produces a larger step along the search direction. This is one of advantages of DFO-TR and e-SLS methods over the 0-LBM method.

In the next section, the study case 2 with nonlinear constraints is analysed.

2.8.3 Case 2 Optimization problem with nonlinear constraints

In the second case, the algorithms are tested on the following optimization problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} \quad (x_1 - 2.7)^2 + 0.5(x_2 - 0.5)^2 - 5 \tag{2.156}$$

$$\text{subject to} \quad 1.5\sin(x_1) - x_2 \leq 0 \tag{2.157}$$

In this problem, the objective function remains the same as in the case 1, while the constraint is changed to a general nonlinear constraint. The optimization result is shown in the following figures. Figure 2.11 shows the optimization trajectories of all algorithms, Figure 2.12 shows the convergence of objective function and Figure 2.13 illustrates the constraint values of all algorithms in the optimization.

For optimization problem with non-linear constraint, one question is how to approach to the optimal point if it is obstructed by non-linear constraint?

The 0-LBM method utilizes the barrier term in the barrier function to perform a sliding optimization trajectory along the boundary towards the optimal point. Because, when decision points are close to a boundary, the barrier term considerably increases the value of barrier function, thus minimizing the barrier function equivalently repels the decision points away from the boundary. By searching for points producing a less value in the barrier function, the optimization trajectory goes along the boundary and approaches to the optimal point. However, the optimization performance of 0-LBM method depends on the choice of barrier coefficient. Ideally, the coefficient shall gradually decreases to zero,

while if the decreasing speed is too large, it approaches to a boundary too early before it is close to the optimal point. On the other hand, if the decreasing speed is too slow, the convergence rate can be slow because the large weighting factor on the barrier term keeps the decision points away from the boundary, which the optimal point is close to.

The e-SLS method uses gradient projection to tackle with non-linear constraint. When the optimization trajectory is close to a boundary below a threshold h, the algorithm projects the search direction on the tangent of the boundary, so that at that iteration, the algorithm searches in the direction parallel to the boundary and towards the optimal point. In this case, the optimization performance depends on the value of h, the less h is, the less optimality gap it is in convergence.

The SafeOpt-MC method is not affected by the shape of constraint function, because it is a global optimization method that it firstly explores possibly the entire decision space before making samples that minimize the objective function.

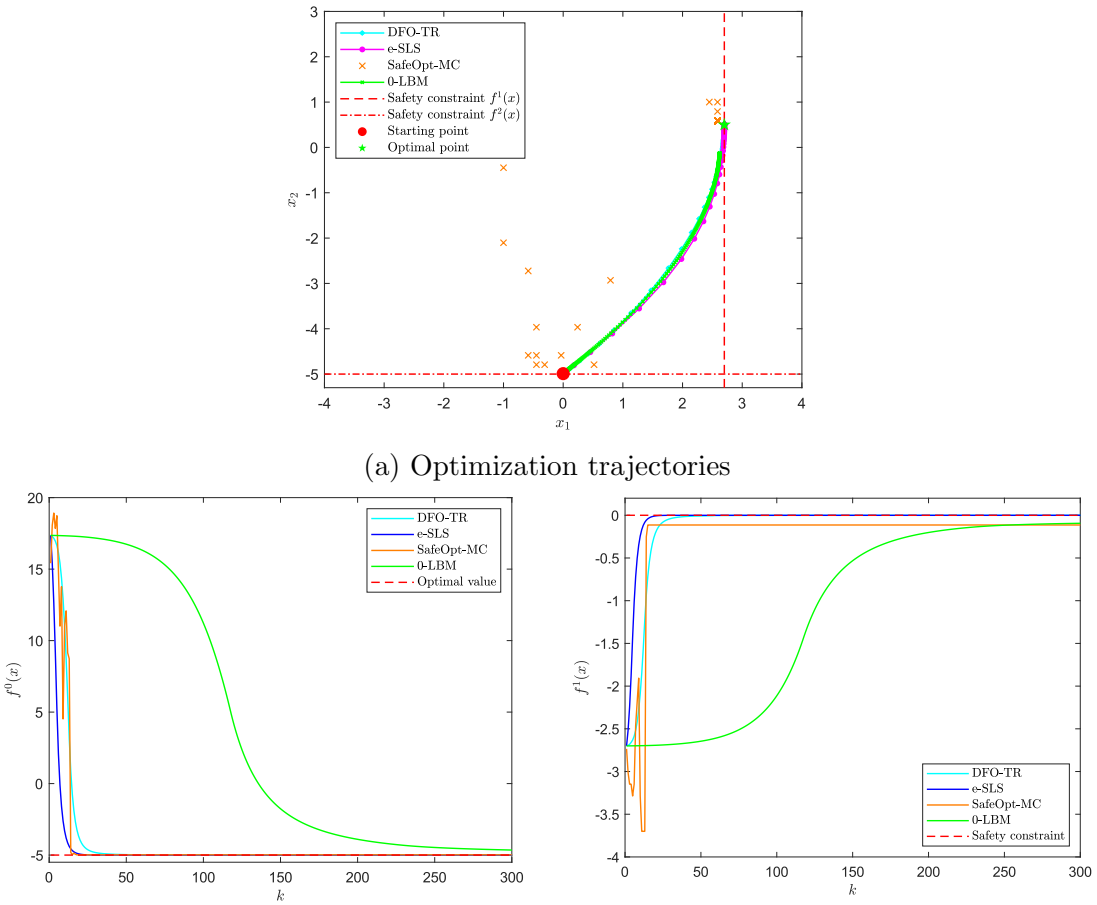
The DFO-TR method is not suitable for non-linear constrained optimization problem, since it minimizes solely the objective function while not considers penalizing being too close to the boundary. Thus, the optimization trajectory can be obstructed by non-linear constraints.

From Figure 2.11 one can see that both e-SLS and 0-LBM method produce sliding optimization trajectories along the non-linear boundary approaching to the optimal point. The SafeOpt-MC method finds the optimal point after some iterations of exploration in the decision space. The DFO-TR method directly goes in the direction towards to the optimal point and it is obstructed by a non-linear boundary between the starting point and the optimal point.

From Figure 2.12, it can be seen that though the e-SLS method and 0-LBM method has similar optimization trajectories, the number of iteration in 0-LBM method is much larger than the values of e-SLS method, as the same in the first simulation case. Both e-SLS and SafeOpt-MC methods converge to the optimal point within 50 iterations.

By Figure 2.13 one can see that the e-SLS, SafeOpt-MC and 0-LBM method are safe during optimization. The DFO-TR method is safe with exact measurements while some measurements of it in the presence of measurement noises are unsafe.

In the next chapter, the study of optimal power flow problem is introduced. This problem generally is a large-scale non-linear, non-convex optimization problem. For system with smaller scale, the problem can be solved with model-based optimization approaches, however, for some power systems, the exact model of it is either not available or is not accurate. In this case, model-free optimization techniques are used for solving the problem. After introducing the optimal power flow problem, the e-SLS method is implemented for solving the problem and its performance is analysed.

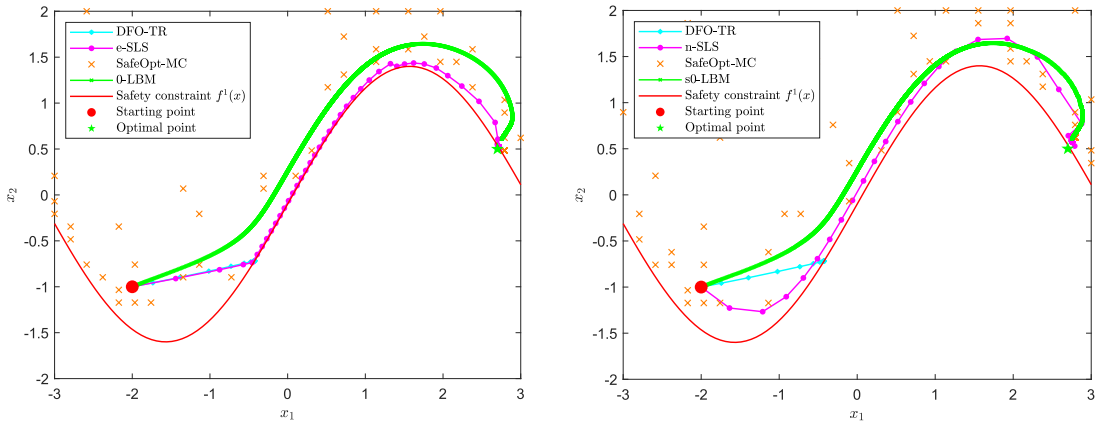


(a) Optimization trajectories

(b) Convergence of objective function

(c) Constraint function values

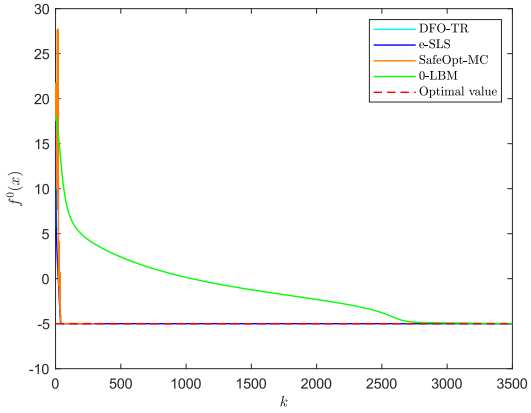
Figure 2.10: Optimization result of different algorithms with exact measurements and starting point at $\boldsymbol{x}_{start} = [0, -4.99]$. Cyan, magenta, green lines represent the result of DFO-TR method, e-SLS method and 0-LBM method respectively. The orange crosses are the sample points of SafeOpt-MC method. (a) shows the optimization trajectories of different algorithms, (b) is the convergence of objective and (c) is the constraint function values.



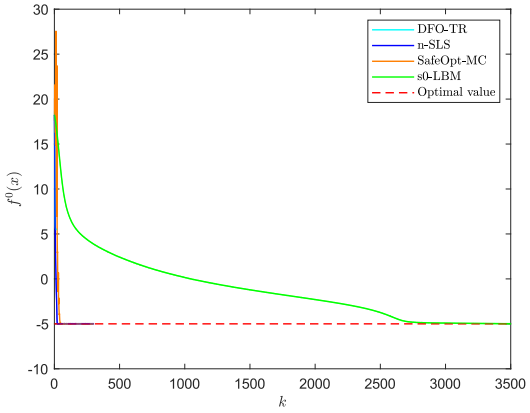
(a) Optimization trajectories of case 2 with exact measurements

(b) Optimization trajectories of case 2 with noisy measurements

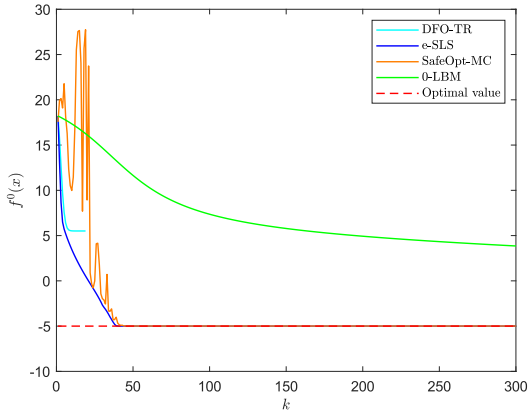
Figure 2.11: (a) Optimization trajectory of different algorithms with exact measurements. The red point denotes the starting point, green star denotes the optimal point, red curve is the non-linear safety constraints. Cyan, magenta, and green lines are the optimization trajectories of DFO-TR method, e-SLS method and 0-LBM method respectively. The orange crosses denote the sample points of SafeOpt-MC method. (b) Optimization trajectories of different algorithms with measurement noise $\xi \sim \mathcal{N}(0, 0.01^2)$.



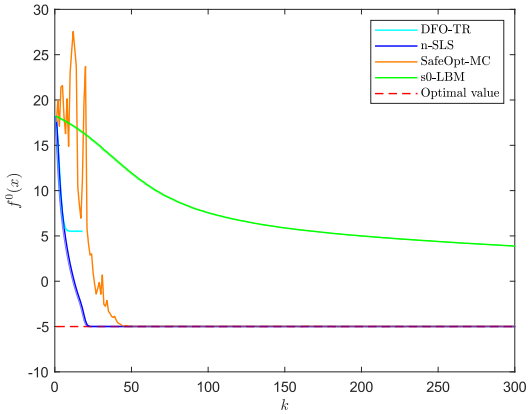
(a) Convergence of objective function of case 2 with exact measurements



(b) Convergence of objective function of case 2 with noisy measurements

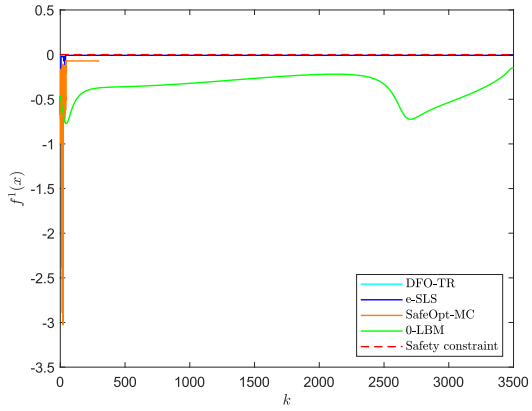


(c) The first 300 iterations of (a)

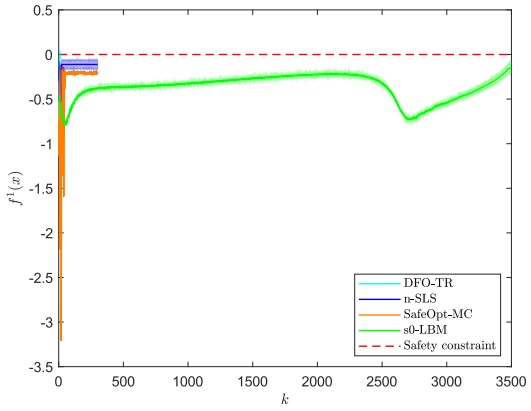


(d) The first 300 iterations of (b)

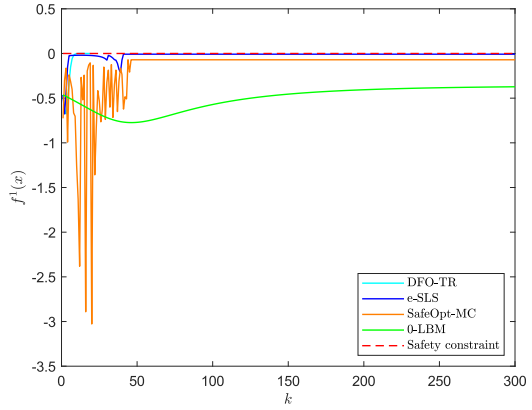
Figure 2.12: (a) Convergence of objective function with exact measurements. Cyan, blue, orange and green lines are the convergence result of DFO-TR method, e-SLS method, SafeOpt-MC method and 0-LBM method respectively. The red dash line is the ground true optimal value. (b) Convergence of objective function with measurement noise $\xi \sim \mathcal{N}(0, 0.01^2)$. (c) and (d) The first 300 iterations of optimization with exact measurements and noisy measurements respectively.



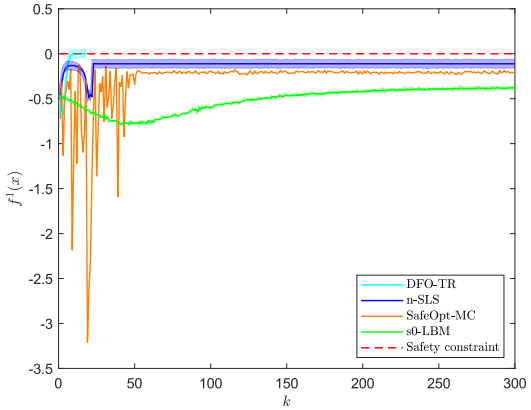
(a) Constraint function values of case 2 with exact measurements



(b) Constraint function values of case 2 with noisy measurements



(c) The first 300 iterations of (a)



(d) The first 300 iterations of (a)

Figure 2.13: (a) Constraint function values with exact measurements. Cyan, blue, orange and green lines are the convergence result of DFO-TR method, e-SLS method, SafeOpt-MC method and 0-LBM method respectively. The red dash line is the constraint upper bound. (b) Constraint function values with measurement noise $\xi \sim \mathcal{N}(0, 0.01^2)$. (c) and (d) The first 300 iterations of optimization with exact measurements and noisy measurements respectively.

3 Optimal Power Flow Problem

3.1 Introduction

Nowadays, electric energy has become one of the most important energy sources for our industrial production and daily life. The electric energy in every factories and households is supplied by the electric power systems.

A power system generally involves the following operations: electricity generation, electricity transmission and electricity distribution. For economic reasons, the generation stations are usually set on remote regions close to the areas where the primary energy, such as petroleum, coal, natural gas, or other renewable energy, is supplied. The produced electric power from the generation stations is transmitted by long distance to the local substations via transmission lines and eventually distributed to the nearby consumers.

Efficiency and operational cost of power systems is one of the key concerns of the power system researcher and engineer communities. Currently, there are several open questions catching researcher's attention. Firstly, the long-distance transmission inevitably causes energy losses due to the resistance of the transmission lines. One nature question is that how can the transmission losses of the power system be minimized? Secondly, the power loads on different areas may considerably diverse from each others, for example, the power demand for industrial usage can be a few orders of magnitude lager than the civil usage. How can the power generation capacities be reasonably assigned to various load centers with different power demands to increase the total efficiency of the power system becomes a question. These problems are one of many instances of optimization applications in the area of power engineering and they have been studied for the last decades in research and industry. A fundamental problem called the optimal power flow (OPF) problem is the common formulation for this study. By setting different objective functions of the OPF problem, the focus of solving the optimization problem can be shifted for various engineering needs.

There exist many approaches for solving the OPF problems, however, theses methods are based on a prerequisite that the model of the target power system is given and accurate. However, for many power systems, the exact model may be not available, due the reasons that the the power system has been built for a long period of time such that the model of the system is lost for some reasons, or the model can not present the real system because some parameters of the system

have been changed in the past, for example, the impedance of the power transmission lines may vary due to ageing or temperature change. In such cases, the conventional model based techniques may fail to provide a good solution given an inaccurate model of the target power system, or even not applicable without any model for the system in hand.

Therefore, the development of model-free optimization techniques come into researcher's mind. Implementation of model-free optimization in solving optimal power flow problem is one of the topics researchers and engineers discussed. In this thesis, the implementation of proposed safe line search algorithm on solving the optimal power flow problem is studied.

This chapter consists of the following sections: Section 3.2 introduces the optimal power flow problem definition. Section 3.3 describes the formulation of AC power flow problem. Section 3.4 presents the implementation of the safe-line-search algorithm on the AC power flow problem.

3.2 Optimal power flow problem definition

Optimal power flow (OPF) problem is the problem for supplying electricity to all loads in the power system, satisfying individual requirements of the load centers and keeping at the minimum possible cost. To solve this problem, a set of information is given:

- The electric demands of every loads in the power system;
- The information about generators: generation capacity, cost per unity energy produced by each generator, etc.;
- The parameters of the transmission network, e.g., impedance, charging susceptance.

Having the information about the power system, the goal of the optimal power flow is to determine which and how much electricity to dispatch by each generator such that, while satisfying various constraints, all electric demands are met at the minimum cost.

In the most completed setting, the OPF problem is a non-linear, non-convex optimization problem. The problem is formulated on the basis of AC power flow equations in a three-phase power system, which serves as the equality constraints in the optimization problem. Along with the power flow equation constraints, a set of operational constraints such as generator capacities, generator and load voltage bounds and line power flow limits are considered in the problem.

Concerning different requirements and constraints, the optimal power flow problem can be classified into three classes:

- The Economic Dispatch problem
- The DC Optimal Power Flow (DC-OPF) problem
- The AC Optimal Power Flow (AC-OPF) problem

The term "Economic Dispatch" has been used to describe different variations of the standard optimal power flow problem. While, the most commonly used definition in the literature is as follows:

Definition 6 (**Economic Dispatch**)**.** *The optimization process of minimizing the generation cost while keeping the satisfaction of various electric power demands, by determining the operation of generators, given the total power demands, and the minimum and maximum possible generation limits of each generator.*

The economic dispatch problem is the fundamental problem of the optimal power flow. A set of constraints such as generator and load voltage constraints, line power flow constraints are neglected in this optimization problem, while they are important in the practical operation of power systems. These additional constraints to the economic dispatch are involved in the more general optimal power flow problem setting. The economic dispatch problem itself can be solved purely graphically using a merit-order curve without any optimization procedures. A more restricted and realistic optimal power flow problem is the DC Optimal Power Flow (DC-OPF) problem.

The DC-OPF is the optimal power flow problem using a linearized version of the original non-linear AC power flow equations in the power system. In the DC-OPF problem, in addition to the setting in the economic dispatch problem, the line power flow constraints are considered. These constraints restrict that the power flowing in the transmission lines shall not exceed the predefined limits. In the DC-OPF problem, the constraints on the power flow limits, however, only account for the real power while the reactive power limits are neglected due to the linearization of the AC-power flow equations. The DC-OPF problem is a convex problem, thus several advantages can be expected such as: a global optimum solution can be guaranteed; it can be solved with a fast convergence rate. The more practical representation of the optimal power flow problem is the AC-OPF problem.

The AC Optimal Power flow problem is the optimization problem considering the complete AC power flow equations and the full set of operational constraints. Compare to DC-OPF, AC-OPF is more accurate to the practical system, and it considers the generator and load voltages constraints, the reactive power flow constraints and the transmission losses. However, the full consideration of optimal power flow problem is a non-linear, non-convex optimization problem, which is harder to be solved and there is no guarantee that a global optimum can be found.

As the economic dispatch problem and the DC-OPF problem have been solved with mature techniques, in this thesis, the main focus is on solving the AC-OPF problem.

Before solving the AC-OPF problem, an introduction and some descriptions about the power system modelling are given. This modelling of power systems is accurate and reliable if the exact model for the power system to be analysed exists. And if accurate model for the target system is available, the first principle for solving the optimization problem is to use model-based methods.

3.3 AC Optimal Power Flow Problem

The AC Optimal Power Flow problem considers the complete non-linear AC power flow equations and the full set of operational constraints. The goal of the optimization problem is to minimize the total generation cost of the power system while satisfying various constraints. The AC Optimal Power Flow problem is formulated below:

Problem 4. ***AC Optimal Power Flow Problem***

variables:

$$P_k^g,\ \forall k \in G \tag{3.1a}$$
$$Q_k^g,\ \forall k \in G \tag{3.1b}$$
$$|V_i|,\ \forall i \in N \tag{3.1c}$$
$$\delta_i,\ \forall i \in N \tag{3.1d}$$

minimize:

$$\sum_{k \in G} c_{2k} {P_k^g}^2 + c_{1k} P_k^g + c_{0k} \tag{3.1e}$$

subject to:

$$S_i^g - S_i^d = |V_i| \sum_{(l,i,j) \in E} |Y_{ij}||V_j| \angle(\delta_i - \theta_{ij} - \delta_j),\ \forall i \in N \tag{3.1f}$$

$$S_{ij} = \left({y_{ij}^s}^* - j \frac{b_{ij}^{sh}}{2} \right) \frac{1}{\tau_l^2} |V_i|^2 - \frac{{y_{ij}^s}^*}{\tau_l e^{j\theta_l^t}} V_i V_j^*,\ \forall (l,i,j) \in E \tag{3.1g}$$

$$S_{ji} = \left({y_{ji}^s}^* - j \frac{b_{ji}^{sh}}{2} \right) \frac{1}{\tau_l^2} |V_j|^2 - \frac{{y_{ji}^s}^*}{\tau_l e^{-j\theta_l^t}} V_j V_i^*,\ \forall (l,i,j) \in E \tag{3.1h}$$

$$\delta_{ref} = 0 \tag{3.1i}$$
$$(P_k^g)^{lb} \leq P_k^g \leq (P_k^g)^{ub},\ \forall k \in G \tag{3.1j}$$
$$(Q_k^g)^{lb} \leq Q_k^g \leq (Q_k^g)^{ub},\ \forall k \in G \tag{3.1k}$$
$$|V_i|^{lb} \leq |V_i| \leq |V_i|^{ub},\ \forall i \in N \tag{3.1l}$$
$$|S_{ij}| \leq |S_{ij}|^{ub},\ \forall (l,i,j) \in E \tag{3.1m}$$
$$|S_{ji}| \leq |S_{ji}|^{ub},\ \forall (l,i,j) \in E \tag{3.1n}$$

where:

- P_k^g is the real power generation of generator k.
- Q_k^g is the reactive power generation of generator k.
- $|V_i|$ is the magnitude of complex voltage at bus i.
- δ_i is the phase angle of complex voltage at bus i.
- Objective function 3.1 represents the total generation cost of the power

system.

- Constraints 3.1f are the AC power flow equations.
- Constraints 3.1g and 3.1h represent the Ohm's Law.
- Constraint 3.1i sets the voltage angle at the reference bus to zero.
- Constraint 3.1j and 3.1k bound the real and reactive power generations of all generators in the power system.
- Constraints 3.1l bound the voltage magnitudes at all buses in the power system.
- Constraints 3.1m and 3.1n bound the branch power flows.

The objective function 3.1 is a polynomial cost function of real power generations of all generators. Coefficients c_{2k}, c_{1k}, c_{0k} are the weighting factors for the second, first and zero-th order terms in the polynomial, and they can vary for different generators to assign different weighting of importance for each generator.

The difficulties of solving the AC-OPF problem come from the facts that, firstly, the multiplications of bus voltages in the equality constraints make the the problem fully non-linear and non-convex. Secondly, for large-scale power system, the number of buses can grow up to thousands, which make the number of decision variables and constraints be a large value.

The problem is harder if there is not an exact model for a large-scale power system. In this case, a model-free optimization method shall be implemented for solving the AC-OPF problem. And the requirement of the scalability shall be met by the optimization algorithms.

As analysed in section 3.3, the DFO-TR method faces difficulties in solving problems with non-linear constraints, the 0-LBM method requires a large number of iterations and the optimality gap is is significant when the optimal point occurs at a boundary. The SafeOpt-MC method provides a property of global optimization, while it is hard to be scaled for sovling problems with high dimensions, such as the AC-OPF problem. The e-SLS method has a property of scalability and rather fast convergence rate. Therefore, it is a suitable choice using the e-SLS method for solving the AC-OPF problem.

In the next section, the proposed safe line search algorithms are implemented on solving the AC-OPF problem in a black-box setting via simulation.

3.4 Implementation of safe line search algorithms for AC-OPF problem

3.4.1 Problem setup

In this section, the AC-OPF problem is treated as black-box optimization problem, which means that the exact model of the power system is not available to the optimization algorithms, but only the function measurements are accessible.

The AC-OPF problem is solved via simulation with a toolbox MATPOWER on several benchmark test systems described below. The MATPOWER toolbox [ZMT11], [ZM20] is an open-source tool for electric power system simulation and optimization. For a given configuration of a power system, the *runpf* function embedded in the MATPOWER tools can compute the values of all variables in the power system fulfilling the equality constraints imposed by the AC-power flow equations. While all inequality constraints are not considered by the *runpf* function. Thus, given a set of decision variables as input to the MATPOWER simulation environment, the optimization algorithms take measurements from the MATPOWER's outputs at each iteration. The algorithms attempt to determine the decision point that minimizes the total generation cost of the power system, while fulfilling all safety constraints according to the obtained measurements. This is the case in the real applications, where feeding a set of decision variables to a power system, one can measure the variables on every buses of the power system and use them as data for optimization algorithms. This setting is a therefore a model-free safe optimization problem.

The optimization results of model-free methods are compared to the results computed by a model-based method, which can be treated as a benchmark result of the optimization problem. The benchmark result can be obtained by using the *runopf* command embedded in the MATPOWER tools, by which it solves the optimal power flow problem with standard model-based optimization methods.

The power system configurations used in the simulation are from the Power Grid Library (PGLib-OPF)[1] proposed by [Bab+19]. The PGLib-OPF is developed for benchmarking AC-OPF algorithms. The library includes more than 60 power system test cases ranging from a simple 3 bus system to a 30000 bus system. For each test case, the generation, loading and network data are configured and a benchmark result is provided for reference. As in our model-free optimization setting, an initial feasible decision point is required, while some test cases with default configuration are not initially feasible, thus, in the simulation, some configurations of the test cases are modified so that an initial feasible point is available.

To demonstrate how an optimal power flow problem is solved, a detailed de-

[1]The repository can be found at `https://github.com/power-grid-lib/pglib-opf`

scription of the optimization process on an IEEE-30 bus system with the proposed e-SLS algorithm is firstly presented, and the optimization result is compared to that of other model-free methods including the DFO-TR method and 0-LBM method. Then, a brief summary of optimization results of other test cases with the e-SLS method is given at the end of this section.

3.4.2 Optimization of IEEE-30 bus system

An example test system is the IEEE-30 bus system. In this system, the number of buses is 30, the number of generator buses including slack bus is 6. The decision variables are the real powers on 5 generator buses, excluding the one of slack bus, and the bus voltage magnitudes on all 6 generator buses. Thus, in total 11 decision variables. The constraint consists of equality constraints of the AC-power flow equations and the voltage angle on the slack bus, and inequality constraints including the operational limits on the bus voltage magnitudes, real and reactive power generations of all generators and apparent power flows on all branches. These constraints can be safety-critical, for example, violating the allowable maximum power flow on branches can lead to damage in the power transmission lines and affect the entire power system, even result in human injury. Thus, these inequality constraints can be seen as safety-critical. For the IEEE-30 bus system, there are in total 158 inequality constraints. The objective function in this simulation is the total generation cost of the power system, which is computed from all real power generations on every generator buses.

The single-line diagram of the IEEE-30 bus system is shown in Figure 3.1. Table 3.1 shows the bus data of the IEEE-30 bus system before optimization. The base voltage for all buses in this power system is $V_{base} = 135kV$. Table 3.2 shows the branch data before optimization. The limit is the upper bound on the apparent power flow at each branch. Table 3.3 shows the generator data of the power system before optimization. The PG limits and QG limits are the corresponding upper and lower bounds on the real and reactive power generations of each generator. The VG limits are bounds on the generator voltage magnitudes. Table 3.4 shows the polynomial coefficients of the objective function.

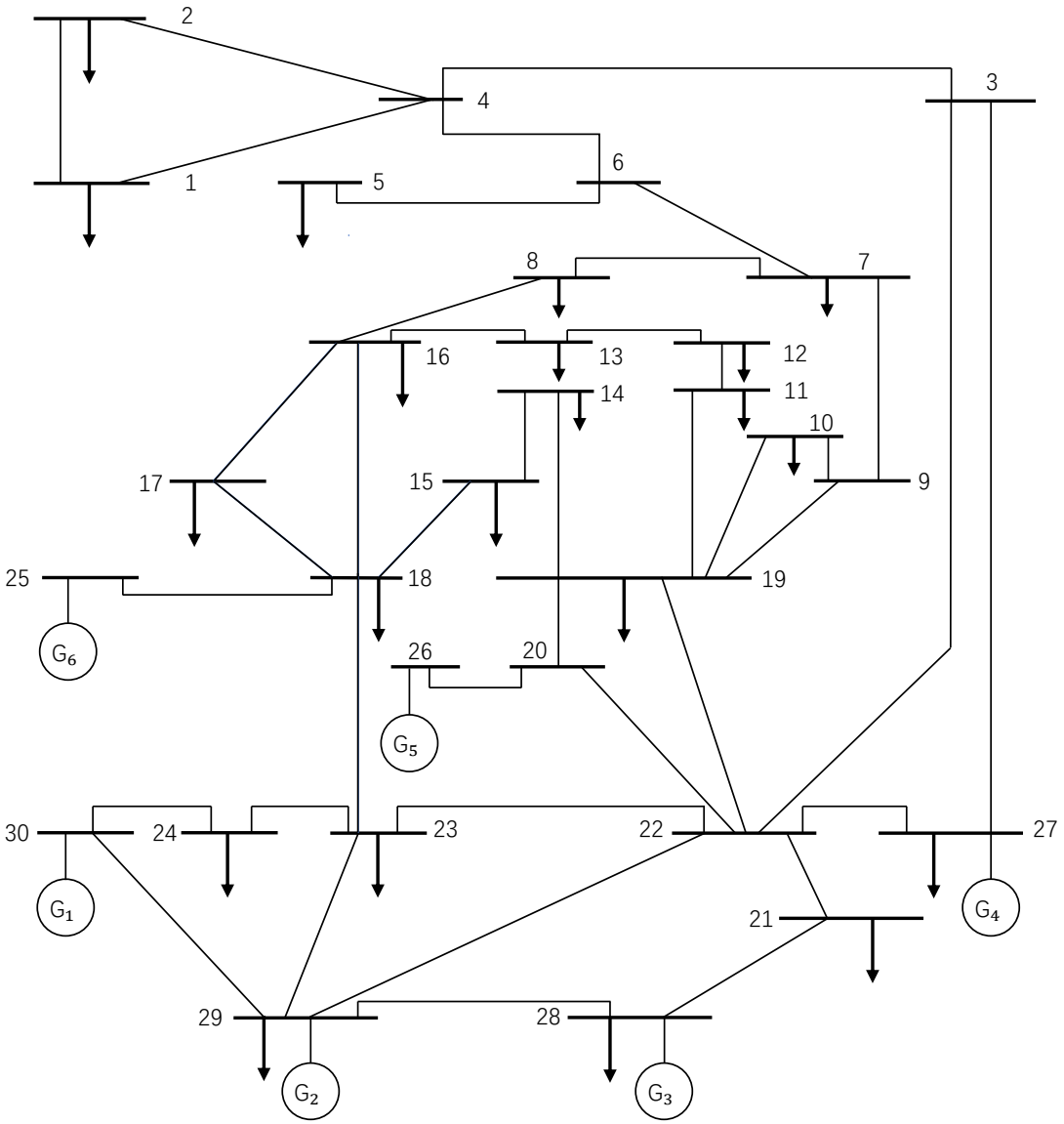


Figure 3.1: Single-line diagram of IEEE-30 bus system

Table 3.1: Bus data of IEEE-30 bus system before optimization

Bus No.	Bus type	PD (MW)	QD (MVAr)	VM (p.u.)	VA (degrees)
1	Slack	0.00	0.00	1.00	0.00
2	PV	21.70	12.70	1.03	-3.05
3	PQ	2.40	1.20	1.00	-3.73
4	PQ	7.60	1.60	1.00	-4.46
5	PQ	94.20	19.00	1.00	-8.06
6	PQ	0.00	0.00	1.00	-5.16
7	PQ	22.80	10.90	1.00	-6.88
8	PQ	30.00	30.00	1.00	-5.06
9	PQ	0.00	0.00	1.01	-5.13
10	PQ	5.80	2.00	1.00	-7.40
11	PQ	0.00	0.00	1.05	-1.03
12	PQ	11.20	7.50	1.01	-6.02
13	PV	0.00	0.00	1.03	-3.17
14	PQ	6.20	1.60	0.99	-7.14
15	PQ	8.20	2.50	0.99	-7.45
16	PQ	3.50	1.80	1.00	-6.91
17	PQ	9.00	5.80	0.99	-7.48
18	PQ	3.20	0.90	0.98	-8.19
19	PQ	9.50	3.40	0.98	-8.42
20	PQ	2.20	0.70	0.99	-8.23
21	PQ	17.50	11.20	0.99	-8.06
22	PV	0.00	0.00	1.00	-8.10
23	PV	3.20	1.60	0.99	-8.46
24	PQ	8.70	6.70	1.00	-9.42
25	PQ	0.00	0.00	0.99	-9.34
26	PQ	3.50	2.30	0.97	-9.78
27	PV	0.00	0.00	0.99	-9.03
28	PQ	0.00	0.00	1.00	-5.53
29	PQ	2.40	0.90	0.97	-10.35
30	PQ	10.60	1.90	0.96	-11.29

$V_{base} = 135kV$

Table 3.2: Branch data of IEEE-30 bus system before optimization

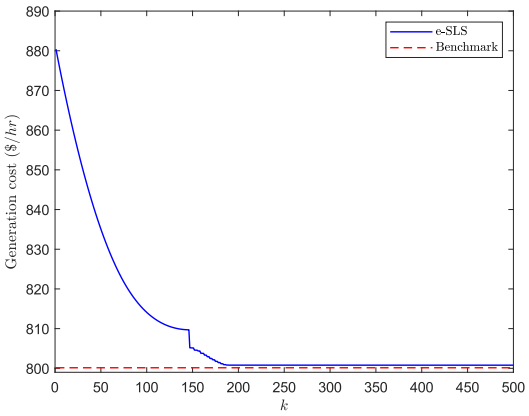
Branch No.	"from" Bus	"to" Bus	"from" bus injection P(MW)+iQ(MVAr)	"to" bus injection P(MW)+iQ(MVAr)	Limit (MVA)
1	1	2	72.94-66.63i	-71.1+69.44i	600
2	1	3	33.06-10.08i	-32.53+10.21i	600
3	2	4	16.9+5.87i	-16.72-7.22i	600
4	3	4	30.13-11.41i	-29.99+11.38i	600
5	2	5	45.82+0.78i	-44.88+1.03i	600
6	2	6	23.36+4.54i	-23.04-5.5i	600
7	4	6	28.38-4.85i	-28.29+4.74i	600
8	5	7	-12.64+12.47i	12.79-13.11i	600
9	6	7	35.93-2i	-35.59+2.21i	600
10	6	8	-2.76+5.42i	2.76-5.85i	600
11	6	9	-0.25-5.1i	0.25+5.16i	600
12	6	10	7.04+0.35i	-7.04-0.07i	600
13	9	11	-36.68-16.71i	36.68+20i	600
14	9	10	36.43+11.55i	-36.43-9.99i	600
15	4	12	10.73-0.91i	-10.73+1.2i	600
16	12	13	-36.68-12.03i	36.68+14.08i	600
17	12	14	8.07+0.93i	-7.99-0.76i	600
18	12	15	19.78+1.26i	-19.53-0.76i	600
19	12	16	8.35+1.13i	-8.29-0.99i	600
20	14	15	1.79-0.84i	-1.78+0.84i	600
21	16	17	4.79-0.81i	-4.77+0.86i	600
22	15	18	6.38+1.24i	-6.33-1.15i	600
23	18	19	3.13+0.25i	-3.12-0.23i	600
24	19	20	-6.38-3.17i	6.4+3.2i	600
25	10	20	8.68+4.09i	-8.6-3.9i	600
26	10	17	4.25+6.71i	-4.23-6.66i	600
27	10	21	16.62+2.72i	-16.52-2.51i	600
28	10	22	8.13-0.18i	-8.08+0.27i	600
29	21	22	-0.98-8.69i	0.99+8.71i	600
30	15	23	6.73-3.83i	-6.67+3.95i	600
31	22	24	7.09-8.99i	-6.93+9.22i	600
32	23	24	3.47-5.55i	-3.42+5.67i	600
33	24	25	1.65+3.63i	-1.62-3.58i	600
34	25	26	3.55+2.37i	-3.5-2.3i	600
35	25	27	-1.93+1.21i	1.93-1.2i	600
36	28	27	15.24+3.13i	-15.24-2.17i	600
37	27	29	6.2+1.69i	-6.11-1.51i	600
38	27	30	7.1+1.68i	-6.93-1.36i	600
39	29	30	3.71+0.61i	-3.67-0.54i	600
40	8	28	3.92-1.65i	-3.91-0.46i	600
41	6	28	11.35+2.1i	-11.33-2.67i	600

Table 3.3: Generator data of IEEE-30 bus system before optimization

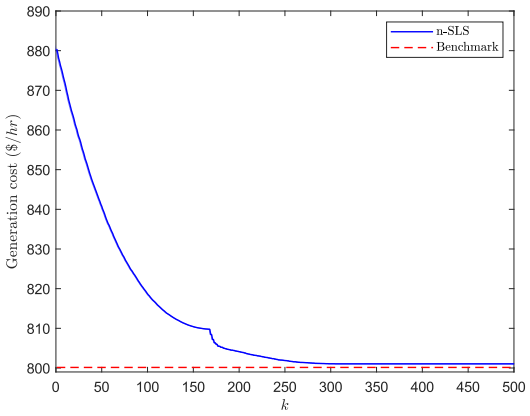
Generator No.	Bus No.	PG (MW)	QG (MW)	PG limits	QG limits	VG (p.u.)	VG limits
1	1	106.00	-76.72	[-300, 300]	[50, 300]	1.00	[0.9, 1.1]
2	2	36.68	93.34	[-300, 300]	[20, 300]	1.03	[0.9, 1.1]
3	5	36.68	32.50	[-300, 300]	[15, 50]	1.03	[0.9, 1.1]
4	8	36.68	22.50	[-300, 300]	[10, 35]	1.03	[0.9, 1.1]
5	11	36.68	20.00	[-300, 300]	[10, 30]	1.03	[0.9, 1.1]
6	13	36.68	14.08	[-300, 300]	[12, 40]	1.03	[0.9, 1.1]

Table 3.4: Polynomial coefficients of the objective function

Generator No.	Bus No.	Coefficients		
		c_2	c_1	c_0
1	1	$3.75e^{-3}$	2.00	0.00
2	2	$1.75e^{-2}$	1.75	0.00
3	5	$6.25e^{-2}$	1.00	0.00
4	8	8.343^{-3}	3.25	0.00
5	11	$2.50e^{-2}$	3.00	0.00
6	13	$2.50e^{-2}$	3.00	0.00



(a) Convergence of total generation cost of IEEE-30 bus system with exact measurements



(b) Convergence of total generation cost of IEEE-30 bus system with noisy measurements

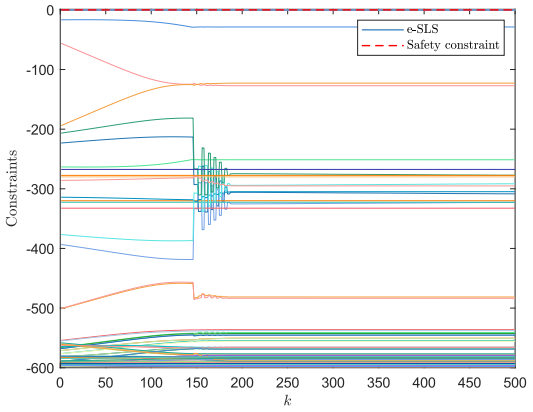
Figure 3.2: (a) Convergence of total generation cost of IEEE-30 bus system with exact measurements. The blue line is the convergence of objective function using the e-SLS method, the red dash line is the benchmark result. (b) Convergence of total generation cost of IEEE-30 bus system with measurement noise $\xi \sim \mathcal{N}(0, 0.01^2)$.

Figure 3.2 shows the convergence of the objective function using e-SLS method. Figure 3.3 shows the constraint function values. From Figure 3.2, it can be seen that the e-SLS method can minimize the total generation cost with only a mild optimality gap to the benchmark result. In the presence of measurement noise, the n-SLS performs a slightly larger optimality gap than the result with exact measurements, while it maintains good convergence to the true optimum.

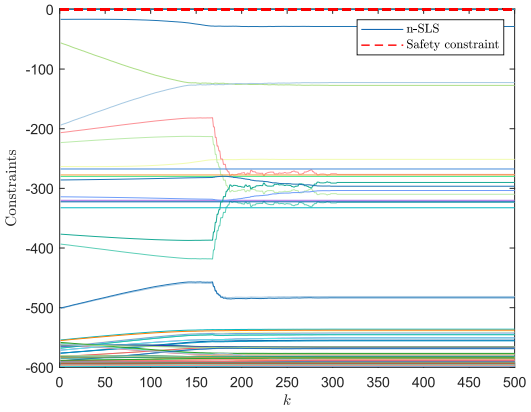
From Figure 3.3, one can see that all 158 inequality constraints are satisfied during optimization process. For the case with measurement noise, the constraint values are more conservative than the case without measurement noises to prevent from violating the constraints. As result, the optimality gap is larger than the case with exact measurements.

By this simulation, the e-SLS and n-SLS methods are proved to perform safe optimization for the optimal power flow problem without explicit knowledge about the underlying power system.

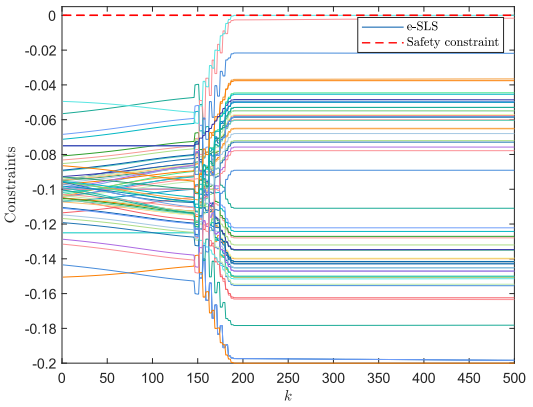
Tables 3.5, 3.6, 3.7 show the bus data, the branch data and the generator data of the IEEE-30 bus system after optimization respectively.



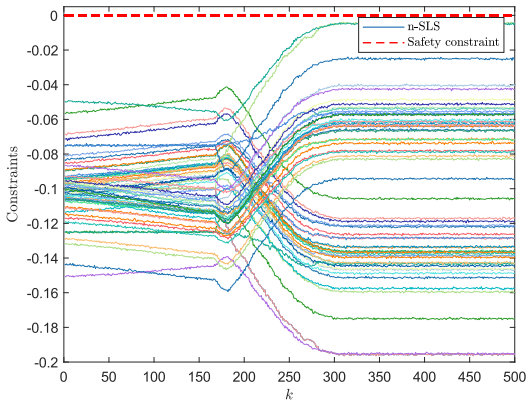
(a) Constraint function values of IEEE-30 bus system with exact measurements



(b) Constraint function values of IEEE-30 bus system with noisy measurements



(c) The largest constraints in (a)



(d) The largest constraints in (b)

Figure 3.3: (a) Constraint function values with exact measurements. The red dash line is the constraint upper bound. (b) Constraint function values with measurement noise $\xi \sim \mathcal{N}(0, 0.01^2)$. (c) and (d) The largest constraints in optimization using e-SLS and n-SLS methods with exact measurements and noisy measurements respectively.

Table 3.5: Bus data of IEEE-30 bus system after optimization

Bus No.	Bus type	PD (MW)	QD (MVAr)	VM (p.u.)	VA (degrees)
1	Slack	0.00	0.00	1.10	0.00
2	PV	21.70	12.70	1.08	-3.28
3	PQ	2.40	1.20	1.06	-5.13
4	PQ	7.60	1.60	1.06	-6.16
5	PQ	94.20	19.00	1.05	-9.62
6	PQ	0.00	0.00	1.05	-7.23
7	PQ	22.80	10.90	1.04	-8.69
8	PQ	30.00	30.00	1.05	-7.42
9	PQ	0.00	0.00	1.06	-9.36
10	PQ	5.80	2.00	1.05	-11.17
11	PQ	0.00	0.00	1.10	-8.13
12	PQ	11.20	7.50	1.05	-10.28
13	PV	0.00	0.00	1.05	-9.47
14	PQ	6.20	1.60	1.03	-11.24
15	PQ	8.20	2.50	1.03	-11.48
16	PQ	3.50	1.80	1.04	-10.95
17	PQ	9.00	5.80	1.04	-11.31
18	PQ	3.20	0.90	1.03	-12.07
19	PQ	9.50	3.40	1.03	-12.22
20	PQ	2.20	0.70	1.03	-12.02
21	PQ	17.50	11.20	1.04	-11.75
22	PV	0.00	0.00	1.05	-11.78
23	PV	3.20	1.60	1.04	-12.25
24	PQ	8.70	6.70	1.05	-12.91
25	PQ	0.00	0.00	1.04	-12.19
26	PQ	3.50	2.30	1.02	-12.59
27	PV	0.00	0.00	1.04	-11.50
28	PQ	0.00	0.00	1.05	-7.71
29	PQ	2.40	0.90	1.02	-12.69
30	PQ	10.60	1.90	1.01	-13.54

$V_{base} = 135kV$

Table 3.6: Branch data of IEEE-30 bus system after optimization

Branch No.	"from" Bus	"to" Bus	"from" bus injection P(MW)+iQ(MVAr)	"to" bus injection P(MW)+iQ(MVAr)	Limit (MVA)
1	1	2	118.62+0.73i	-116.38+2.85i	600
2	1	3	58.55+7.71i	-57.23-4.71i	600
3	2	4	34.16+2.63i	-33.59-2.96i	600
4	3	4	54.83+3.51i	-54.48-2.97i	600
5	2	5	63.82+2.11i	-62.16+2.49i	600
6	2	6	45.39+2.47i	-44.35-1.44i	600
7	4	6	49.51-1.05i	-49.25+1.46i	600
8	5	7	-10.69+11.01i	10.79-11.87i	600
9	6	7	33.87-1.04i	-33.59+0.97i	600
10	6	8	10.38+5.1i	-10.37-5.54i	600
11	6	9	19.95-5.71i	-19.95+6.52i	600
12	6	10	13.63+0.55i	-13.63+0.39i	600
13	9	11	-12.04-19.06i	12.04+20i	600
14	9	10	31.99+12.54i	-31.99-11.39i	600
15	4	12	30.96+5.38i	-30.96-3.11i	600
16	12	13	-11.13-4.63i	11.13+4.82i	600
17	12	14	7.37+0.66i	-7.31-0.53i	600
18	12	15	17.27-0.23i	-17.09+0.58i	600
19	12	16	6.24-0.19i	-6.21+0.26i	600
20	14	15	1.11-1.07i	-1.1+1.08i	600
21	16	17	2.71-2.06i	-2.7+2.08i	600
22	15	18	5.27+0.64i	-5.24-0.58i	600
23	18	19	2.04-0.32i	-2.04+0.32i	600
24	19	20	-7.46-3.72i	7.48+3.77i	600
25	10	20	9.78+4.69i	-9.68-4.47i	600
26	10	17	6.33+7.96i	-6.3-7.88i	600
27	10	21	15.99+2.48i	-15.91-2.3i	600
28	10	22	7.72-0.32i	-7.68+0.4i	600
29	21	22	-1.59-8.9i	1.6+8.92i	600
30	15	23	4.73-4.8i	-4.68+4.89i	600
31	22	24	6.08-9.32i	-5.94+9.52i	600
32	23	24	1.48-6.49i	-1.43+6.6i	600
33	24	25	-1.33+4.99i	1.37-4.91i	600
34	25	26	3.54+2.36i	-3.5-2.3i	600
35	25	27	-4.91+2.54i	4.94-2.49i	600
36	28	27	18.22+2.04i	-18.22-0.83i	600
37	27	29	6.19+1.66i	-6.1-1.5i	600
38	27	30	7.09+1.65i	-6.93-1.36i	600
39	29	30	3.7+0.6i	-3.67-0.54i	600
40	8	28	2.49-1.96i	-2.48-0.38i	600
41	6	28	15.77+1.08i	-15.73-1.66i	600

Table 3.7: Generator data of IEEE-30 bus system after optimization

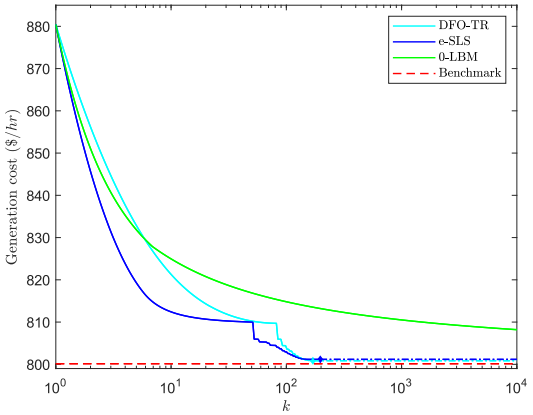
Generator No.	Bus No.	PG (MW)	QG (MW)	PG limits	QG limits	VG (p.u.)	VG limits
1	1	177.17	8.44	[-300, 300]	[50, 300]	1.10	[0.9, 1.1]
2	2	48.69	22.76	[-300, 300]	[20, 300]	1.08	[0.9, 1.1]
3	5	21.35	32.50	[-300, 300]	[15, 50]	1.03	[0.9, 1.1]
4	8	22.12	22.50	[-300, 300]	[10, 35]	1.03	[0.9, 1.1]
5	11	12.04	20.00	[-300, 300]	[10, 30]	1.03	[0.9, 1.1]
6	13	11.13	4.82	[-300, 300]	[12, 40]	1.05	[0.9, 1.1]

A comparison between the convergences of different model-free optimization methods is shown in Figure 3.4. In this figure, the iteration axis is in a logarithmic scale, and it can be seen that the DFO-TR method and e-SLS method have comparable convergence rate, while the 0-LBM method requires two orders of magnitude larger number of iterations to converge and the optimality gap remains significant after 10000 iterations.

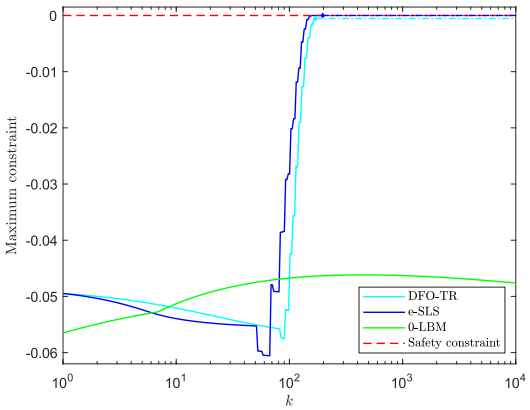
The computation time required by each algorithm to convergence is shown in Table 3.8. In this table, the total time is the time the optimization process finishes and the simulation time counts the time used by simulation of the power system, that is, the time between feeding an input to the power system to obtaining an output from it. From this table, it can be seen that with the DFO-TR method, the majority part of computation time is used by the algorithm itself, while the simulation time accounts only for about 30%. The e-SLS method requires the least amount of computation time and the majority part of time is spent on simulation of the power system. The 0-LBM method has the highest ratio of the simulation time to the total time, which means that the algorithm itself is fast. However, it hardly converges sufficiently to the optimal point, left the largest optimality gap after 10000 iteration. Overall, the e-SLS algorithm has a more balanced performance.

Table 3.8: Computation time of different algorithms for optimization of the IEEE-30 bus system

Algorithms	Number of iterations (T)	Total time (s)	Simulation time (s)	Convergence condition
DFO-TR	**170**	48.968	15.007	ϵ =1e-12
e-SLS	197	**7.836**	**6.507**	ϵ =1e-12
0-LBM	10000	280.149	248.728	T=10000



(a) Convergence of the total generation cost by different algorithms with exact measurements



(b) Maximum constraint of optimization by different algorithms with exact measurements

Figure 3.4: (a) Convergence of total generation cost of the IEEE-30 bus system with exact measurements. Cyan, blue and green lines are the convergence by the DFO-TR method, the e-SLS method and the 0-LBM method respectively. Diamond markers denote the convergent iteration and the followed dash lines present the objective values at the convergent iteration. The red dash line is the benchmark result. The iteration axis is in logarithmic scale. (b) Maximum constraint among all 158 constraints of the IEEE-30 bus system by different algorithms.

With a detailed description of the optimization of AC-OPF problem by an example case of IEEE-30 bus system, the optimization results of other cases by the e-SLS method are briefly summarized on Table 3.9. In this table, the optimality gap is computed via the following formula:

$$OG = \frac{Benchmark - Result}{Benchmark} \times 100\% \tag{3.2}$$

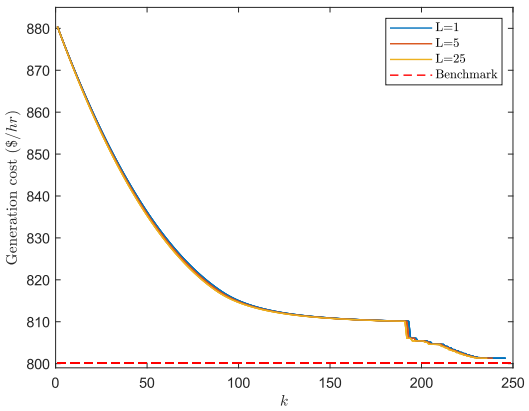
Table 3.9: Optimality gaps on test cases from Power Grid Library

Case Name	Nodes	Branches	Generation cost($/h)		Optimality Gap(%)
			Benchmark	e-SLS	
case9	9	9	5.30e+03	5.30e+03	3.91e-03
pglib_opf_case14	14	20	2.18e+03	2.20e+03	1.55e+00
pglib_opf_case30	30	41	8.00e+02	8.01e+02	8.57e-02
pglib_opf_case57	57	80	9.28e+03	9.44e+03	1.70e+00
pglib_opf_case200	200	245	2.76e+04	3.00e+04	8.98e+00

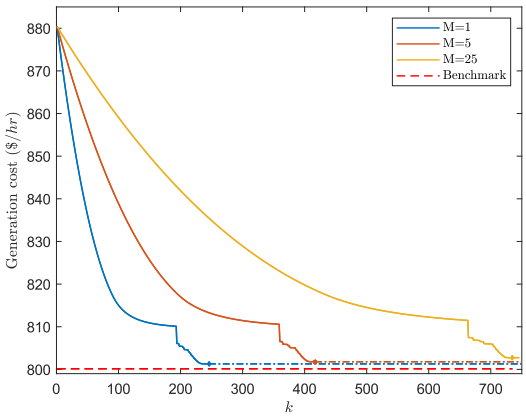
3.4.3 Effect of Lipschitz and smoothness constants

The e-SLS algorithm is formulated based on the assumptions of the Lipschitz and smoothness properties of the underlying functions. The effects of the these constants are analysed.

Figure 3.5 shows the convergence of the total generation cost of the IEEE-30 bus system by the e-SLS algorithm with different Lipschitz and smoothness constants. From Figure 3.5a, it can be seen that, the Lipschitz constant does not affect the convergence result, as it decides only the difference step length used in the finite-difference method for estimating the gradients of the underlying functions. Thus, it is not involved in the actual iteration. However, the convergence depends on the smoothness constant. With a larger M, the number of iterations and the optimality gap increase, since the safe set radius $\mathcal{R}_k^i$ is a function of M, and the larger M is, the smaller $\mathcal{R}_k^i$ will be. From Table 3.10, it shows that the total computation time increases with the smoothness constant M as well.



(a) Convergence of the total generation cost by the e-SLS algorithm with different Lipschitz constants



(b) Convergence of the total generation cost by the e-SLS algorithm with different smoothness constants

Figure 3.5: (a) Convergence of the total generation cost of the IEEE-30 bus system with Lipschitz constants of 1, 5, and 25 respectively. (b) Convergence of the total generation cost of the IEEE-30 bus system with smoothness constants equal to 1, 5, and 25 respectively.

Table 3.10: Computation time of optimization by e-SLS algorithm with different Lipschitz and smoothness constants

L	M	Computation time	Convergence condition
1	1	9.343	ϵ =1e-12
5	1	9.619	ϵ =1e-12
25	1	9.661	ϵ =1e-12
1	5	14.348	ϵ =1e-12
1	25	25.037	ϵ =1e-12

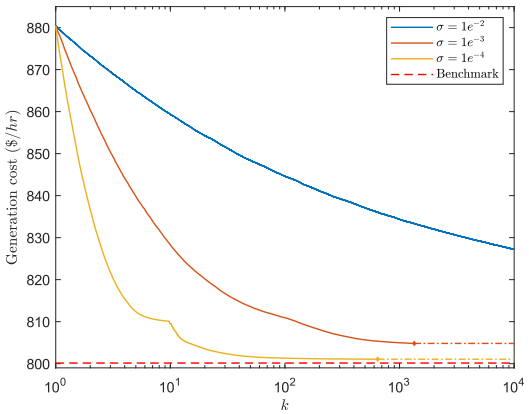
3.4.4 Effect of standard deviation of measurement noises

The standard deviation of measurement noises is critical in safe optimization problem, as the larger the measurement noise is, the more uncertain the gradient estimators of constraints will be. An analysis of the effect of standard deviation of the measurement noises is conducted.

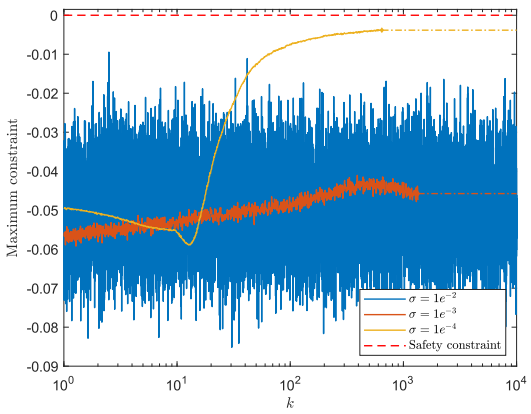
Figure 3.6 and 3.7 illustrate the effect of the standard deviation of the measurement noise on the convergence result. Figure 3.6 shows that, with a larger standard deviation in the measurement noise, the number of iterations considerably increases. As with a larger σ, the estimation deviation $\|\Delta_k^i\|_2$ increases, and with a modification on the gradient estimators, the radius $\mathcal{R}_k^i$ of the safe set decreases to achieve a conservative optimization. Figure 3.6 illustrates the optimization result with different number of iterations for measurement noise with standard deviation of $1e^{-2}$. As the number of measurements increase, the uncertainty in the function values decreases and thus reduce the required iteration numbers. Table 3.11 shows the computation time of the optimization. When the standard deviation $\sigma = 1e^{-2}$ and $n_k = 1$, there are 6 violations during optimization, and they occur in the trials of the step length selection 6. For this challenging setting for safe optimization problem, it is acceptable to have 6 violations out of 10000 iterations.

Table 3.11: Computation time of optimization by n-SLS algorithm with different standard deviations of measurement noises and number of measurements

σ	n_k	computation time (s)	Convergence condition	number of violations
1e-4	1	23.053	ϵ =1e-5	0
1e-3	1	47.093	ϵ =1e-5	0
1e-2	1	653.728	T=10000	6
1e-2	100	50.319	ϵ =1e-5	0
1e-2	1000	63.681	ϵ =1e-5	0

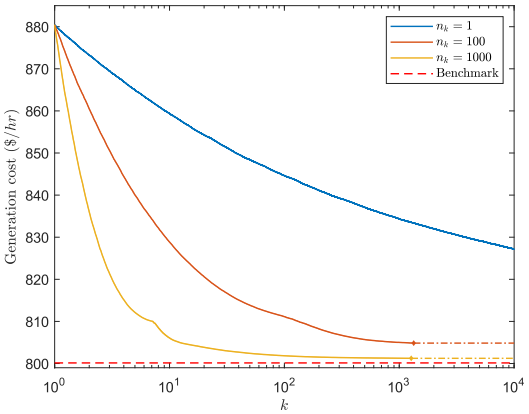


(a) Convergence of the total generation cost with different variances of measurement noises

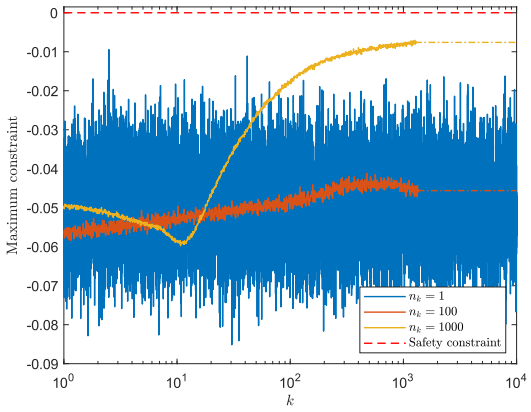


(b) Maximum constraint of optimization with different variances of measurement noises

Figure 3.6: (a) Convergence of the total generation cost of the IEEE-30 bus system with standard deviationss of $1e^{-2}$, $1e^{-3}$,$1e^{-4}$ of the measurement noises.(b) Maximum constraint among all 158 constraints of the IEEE-30 bus system by the n-SLS algorithm with different variances of the measurement noises.



(a) Convergence of the total generation cost with different number of measurements for measurement noises with standard deviation of $1e^{-2}$



(b) Maximum constraint of optimization with different number of measurements for measurement noises with standard deviation of $1e^{-2}$

Figure 3.7: (a) Convergence of the total generation cost of the IEEE-30 bus system with number of measurements of 1, 100, 1000 for measurement noise with standard deviation of $1e^{-2}$. (b) Maximum constraint among all 158 constraints of the IEEE-30 bus system by the n-SLS algorithm with different number of measurements for measurement noise with standard deviation of $1e^{-2}$.

4 Conclusion

In this these, a model-free safe optimization algorithm is proposed. This algorithm does not require explicit knowledge about the objective and constraint functions and is able to converge to a local optimal point of a non-linear, non-convex optimization problem without violating safety constraints.

At each iteration, the proposed algorithm takes measurements of the underlying objective and constraint functions, and estimate their gradients by finite-difference method. With the gradient estimators of the underlying functions, the algorithm determines a descent direction of the objective function as a steepest direction or a Newton-direction, then it constructs a safe set around the current point utilizing smoothness assumptions about the unknown constraint functions. With the determined search direction and the formulated safe set, a line search policy is implemented along the search direction within the area defined by the safe set to find a safe step length that provides a sufficient reduction in the objective function. When the decision point is near a boundary, the search direction is projected onto the perpendicular direction of the gradients of active constraints pointing to the descent direction. By the implementation of search direction projection, the optimization trajectory approaches along the boundary of the feasible set towards the optimal feasible point. The algorithm is safety-guarantee with exact measurements and safe with high probability in case of noisy measurements.

Compare to other model-free optimizations, the proposed algorithm is scalable and has a fast convergence. A set of model-free optimization algorithms are tested on two example problems. The optimization results demonstrate that the proposed algorithm provides a good trade-off between the convergence rate and the computation time and noticeable convergence accuracy.

The proposed algorithm is implemented for solving the large-scale AC Optimal Power Flow problem. It is shown that the algorithm finds optimal points close to the benchmark results without using an explicit model of the target power system. Regarding the parameters of the proposed algorithm, the Lipschitz constant does not affect the convergence rate while the number of iterations and the computation time is inversely proportional to the smoothness constant. With a larger standard deviation in the measurement noises, the safe optimization is more challenging and the algorithm becomes more conservative via reducing the radius of safe sets and thus requires a larger number of iterations and the time of computation .

A Power system modeling

A.1 Single-line diagram

An electric power system is an interconnection network that includes the following components: generators, loads, branches and transformers. In a power system network, generators and loads are mounted on buses[1] of the network. The branches represent the transmission lines in the electric power system and the transformer is set in the middle between two connected buses. An electric power system can be illustrated using a single-line diagram, which presents the components of the power system and the connection relations between each component. In the single-line diagram, generators are represented by letter G, loads are denoted by outgoing arrows and the numbers around buses indicate the bus indices. Figure A.1 shows an example of the single-line diagram of the IEEE 9-bus system.

In the next section, the components of the power system are introduced.

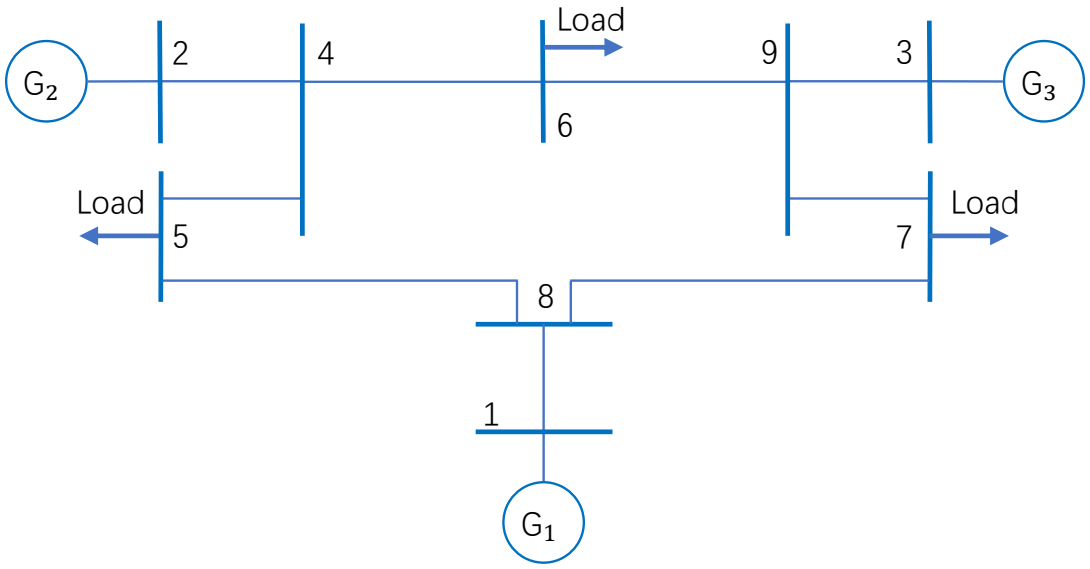


Figure A.1: Single-line diagram of IEEE-9 bus system

[1] A bus is a node on the power system network.

A.2 Components of power system network

In a power system, there exist generally the following components:

- Generators;
- Loads;
- Buses;
- Branches.

For clarity of describing each components, in this section the following notations are used: N denotes the set of bus indices, n_b is the number of buses, G denotes the set of generator indices, n_g is the number of generators, E is the set of branches and n_l is the number of branches.

The first component is generators. In a power system, generators are represented by the real and reactive power productions at buses. For generator $j \in G$, the apparent power generation is:

$$S_j^g = P_j^g + jQ_j^g \tag{A.1}$$

A $n_g \times 1$ vector $\mathbf{S}^g$ is used to denote the power generations of all generators.

The power loads in a power system are modeled as the real and reactive power consumption at buses. For bus $i \in N$, the complex power load is:

$$S_i^d = P_i^d + jQ_i^d \tag{A.2}$$

A $n_b \times 1$ vector $\mathbf{S}^d$ denotes the complex power loads at all buses.

A bus in the power system is the node of the power system network. Generators and loads can be mounted on buses. For a bus $i \in N$, there are four parameters: the bus voltage magnitude $|V_i|$, the bus voltage angle δ_i, the net real power injection P_i^n and the net reactive power injection Q_i^n. The net complex power injection is the difference between the complex power generation and the power demand on a bus. That is:

$$S_i^n = S_i^g - S_i^d \tag{A.3}$$

or:

$$P_i^n = P_i^g - P_i^d \tag{A.4}$$

$$Q_i^n = Q_i^g - Q_i^d \tag{A.5}$$

For each bus, only partial parameters can be specified in advance, and the remaining ones are unknown. According to to the type of known parameters, a bus can be classified as:

- Load bus;
- Generator bus;
- Reference bus.

The load bus is the bus on which only load is connected without any generators. At a load bus $i \in N$, the real and reactive power demand P_i^d and Q_i^d are specified, while the voltage magnitude $|V_i|$ and angle δ_i are unknown. Since the complex power load is specified and the complex power generation is zero, according to A.3, the complex net power injection at the load bus is fully defined by the power load, thus a load bus is also called a PQ bus.

The generator bus is the bus on which a generator is connected. A known and constant load can be possibly connected to a generator bus as well. At a generator bus $j \in G$, the real power generation P_j^g and the bus voltage magnitude $|V_j|$ are specified, while the reactive power generation Q_j^g and the bus voltage angle δ_j remain unknown. Since the real power generation is specified, and the load is either zero or known constant, according to A.4, the net real power injection is fully defined. Thus, along with the known bus voltage magnitude, a generator bus is also called a PV bus.

The reference bus, also known as slack bus, is a special generator bus. At a reference bus $j \in G$, both bus voltage magnitude $|V_j|$ and angle δ_j are given, while the complex power generation S_j^g is unknown. The name of reference bus comes from the fact that the bus voltage angle at this bus serves as a reference for voltage angles at other buses. Conventionally, the voltage angle at the reference bus is set to 0, and there is only one reference bus in a power system.

A generic bus model is illustrated in Figure A.2 and a summary of different types of buses is presented on Table A.1, in which, S stands for "specified" and U denotes "unknown".

Table A.1: Specified and unknown variables of different buses

Bus type	Voltage ($\|V\|\angle\delta$)		Real power (P)			Reactive Power (Q)		
	Magnitude	Angle	Gen.	Load	Net	Gen.	Load	Net
Reference	S	S	U	S	U	U	S	U
Generator/PV	S	U	S	S	S	U	S	U
Load/PQ	U	U	S	S	S	S	S	S

The branches represent the transmission lines in power systems. A branch is modeled by a Π-circuit as shown in Figure A.3. The parameters of a branch $(l, i, j) \in E$ are the series impedance z_{ij}^s between bus i and bus j and two equal

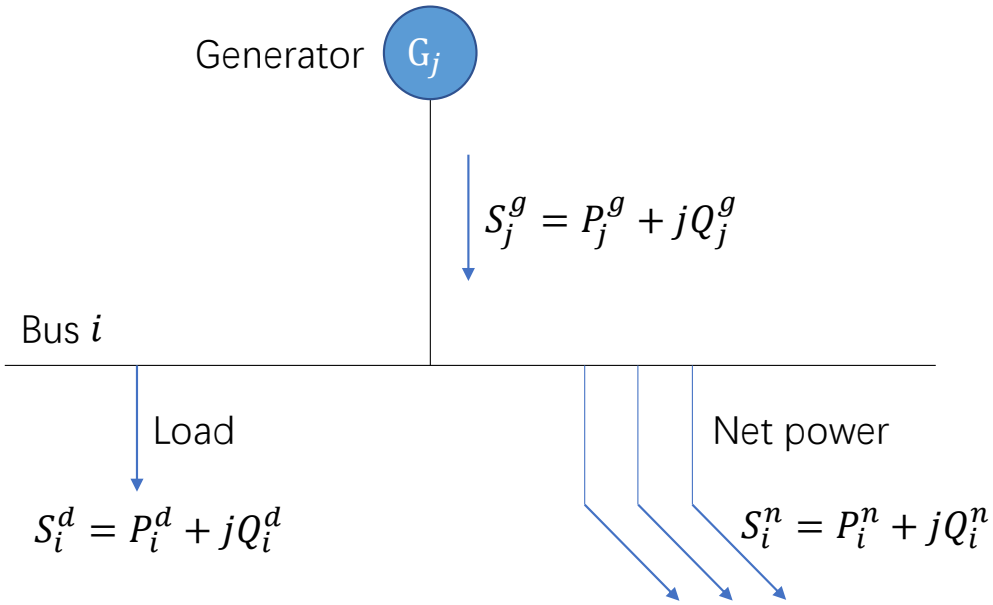


Figure A.2: Genetic bus model

shunt susceptances $\frac{b_{ij}^{sh}}{2}$ and $\frac{b_{ji}^{sh}}{2}$ connected to the two ends of the branch l.

$$z_{ij}^s = r_{ij}^s + x_{ij}^s \tag{A.6}$$

The series impedance is usually converted to series admittance for convenience for computation of bus voltages at two ends of branches:

$$y_{ij}^s = y_{ji}^s = \frac{1}{r_{ij}^s + jx_{ij}^s} \tag{A.7}$$

The equality constraint of AC-OPF problem is the AC power flow equations. These equations connect all components in a power system. In order to formulate the AC power flow equations, the currents at all branches shall be expressed using the four parameters of buses. In the next section, the computation of branch current is studied.

A.3 Computation of branch and bus current

According to the branch model shown in Figure A.3, for a branch $(l, i, j) \in E$, the current flowing from bus i to bus j is:

$$I_{ij} = \frac{1}{N^*}\left(I_{ij}^{sh} + I_{ij}^s\right) = j\frac{b_{ij}^{sh}}{2}\frac{V_i}{\tau_l^2} + y_{ij}^s\left(\frac{V_i}{\tau_l^2} - \frac{V_j}{\tau_l e^{-j\theta_l^t}}\right) \tag{A.8}$$

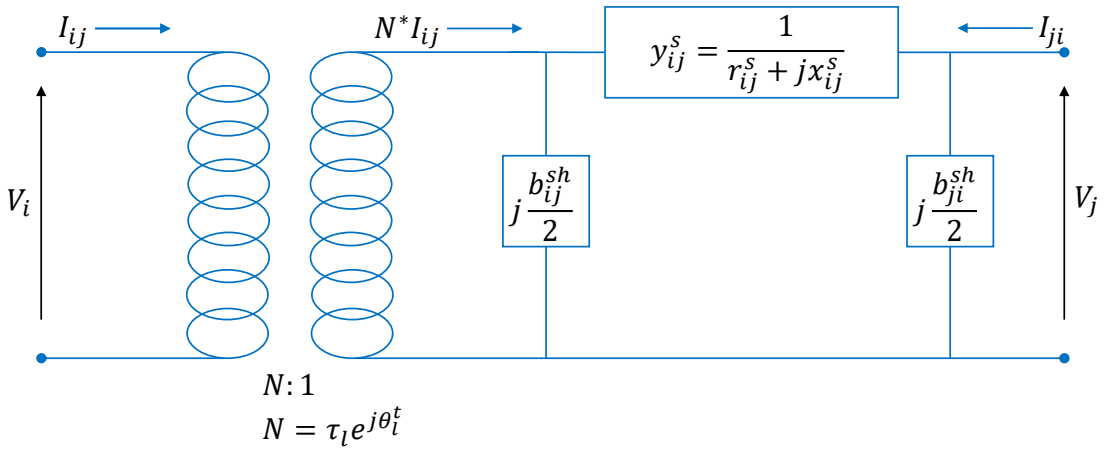


Figure A.3: Π circuit for branch model

written in matrix form as:

$$I_{ij} = \begin{bmatrix} \left(y_{ij}^s + j\frac{b_{ij}^{sh}}{2}\right)\frac{1}{\tau_l^2} & -y_{ij}^s\frac{1}{\tau_l e^{-j\theta_l^t}} \end{bmatrix} \begin{bmatrix} V_i \\ V_j \end{bmatrix} \tag{A.9}$$

Similarly, the current flowing from bus j to bus i is:

$$I_{ji} = I_{ji}^{sh} + I_{ji}^s = j\frac{b_{ji}^{sh}}{2}V_j + y_{ji}^s\left(V_j - \frac{V_i}{\tau_l e^{j\theta_l^t}}\right) \tag{A.10}$$

written in matrix form as:

$$I_{ji} = \begin{bmatrix} -y_{ji}^s\frac{1}{\tau_l e^{j\theta_l^t}} & y_{ji}^s + j\frac{b_{ji}^{sh}}{2} \end{bmatrix} \begin{bmatrix} V_i \\ V_j \end{bmatrix} \tag{A.11}$$

As the current flowing from bus i to j is not equal to the current from bus j to i, the difference between $|I_{ij}|$ and $|I_{ji}|$ represents the power loss on the transmission lines.

Equation A.9 and A.11 can be written in matrix form such that all branch currents can be related with all bus voltages by the following:

$$\mathbf{I}_{br} = \mathbf{Y}_{br}\mathbf{V}_{bus} \tag{A.12}$$

where $\mathbf{I}_{br}$ is a $n_l \times 1$ vector containing currents in all branches and $\mathbf{V}_{bus}$ is a $n_b \times 1$ vector containing all bus voltages in the power system. The $n_l \times n_b$ matrix $\mathbf{Y}_{br}$ that relates the branch currents with the bus voltages is called the branch admittance matrix.

Since the branch current injected at one end of the branch is not equal to the current received at the another end, two branch admittance matrices shall be defined for each branch. Define the end at which the current is flowed out

into a branch as the "*from*" end and the another end as the "*to*" end. Then, the branch admittance matrix computed by Equation A.9 is called the "*from*" branch admittance matrix, denoted by $\mathbf{Y}_{br}^{f}$, and the branch admittance matrix computed by Equation A.11 is called the "*to*" branch admittance matrix, denoted by $\mathbf{Y}_{br}^{t}$.

For a branch $(l,i,j) \in E$, in the "*from*" branch admittance matrix $\mathbf{Y}_{br}^{f}$, the (l,i)-th element is $(y_{ij}^{s} + j\frac{b_{ij}^{sh}}{2})\frac{1}{\tau_l^2}$ and the (l,j)-th element is $-y_{ij}^{s}\frac{1}{\tau_l e^{-j\theta_l^t}}$, other elements in the l-th row of the matrix are zeros. In the "*to*" branch admittance matrix $\mathbf{Y}_{br}^{t}$, the (l,i)-th element is $-y_{ji}^{s}\frac{1}{\tau_l e^{j\theta_l^t}}$ and the (l,j)-th element is $y_{ji}^{s} + j\frac{b_{ji}^{sh}}{2}$, and the remaining elements in the l-th row are zeros.

In order to compute the power flow at each bus, the net current injection at each bus shall be computed. The net current injection at a bus is the sum of currents flowing out from and received at the bus. For bus $i \in N$, the net current injection is:

$$I_i = \sum_{(l,i,m)\in E} I_{im} + \sum_{(l,n,i)\in E} I_{ni} \tag{A.13}$$

The right hand side of Equation A.13 equals to:

$$\sum_{(l,i,m)\in E} I_{im} = V_i \sum_{(l,i,m)\in E} \left(y_{im}^{s} + j\frac{b_{im}^{sh}}{2}\right)\frac{1}{\tau_{l_m}^2} - \sum_{(l,i,m)\in E} y_{im}^{s}\frac{1}{\tau_{l_m} e^{-j\theta_{l_m}^t}} V_m \tag{A.14}$$

and

$$\sum_{(l,n,i)\in E} I_{ni} = V_i \sum_{(l,n,i)\in E} \left(y_{ni}^{s} + j\frac{b_{ni}^{sh}}{2}\right) - \sum_{(l,n,i)\in E} y_{ni}^{s}\frac{1}{\tau_{l_n} e^{j\theta_{l_n}^t}} V_n \tag{A.15}$$

In this thesis, for simplicity, all transformers are assumed to be turned off, that is, the tap ratios N of all transformers are set to 1:

$$N_l = \tau_l e^{j\theta_l^t} = 1,\ \forall (l,i,j) \in E \tag{A.16}$$

In this case, the current injection at each bus i is:

$$I_i = V_i \sum_{(l,i,j)\in E} \left(y_{ij}^{s} + j\frac{b_{ij}^{sh}}{2}\right) - \sum_{(l,i,j)\in E} y_{ij}^{s} V_j \tag{A.17}$$

Similar to the branch admittance matrix $\mathbf{Y}_{br}$, a bus admittance matrix $\mathbf{Y}_{bus}$ is defined to relate all bus net current injections with all bus voltages by:

$$\mathbf{I}_{bus} = \mathbf{Y}_{bus}\mathbf{V}_{bus} \tag{A.18}$$

where $\mathbf{I}_{bus}$ and $\mathbf{V}_{bus}$ are $n_b \times 1$ vectors and $\mathbf{Y}_{bus}$ is a $n_b \times n_b$ matrix.

Denote the (i,j)-th element of bus admittance matrix $\mathbf{Y}_{bus}$ by Y_{ij}. The diagonal elements of matrix $\mathbf{Y}_{bus}$ are:

$$Y_{ii} = \sum_{(l,i,j)\in E} \left(y_{ij}^{s} + j\frac{b_{ij}^{sh}}{2}\right) \tag{A.19}$$

and the off-diagonal elements are:

$$Y_{ij} = -y_{ij}^s \tag{A.20}$$

If there is no connection between bus i and bus j, then $Y_{ij}^s = 0$. Using Equation A.19 and A.20, Equation A.17 can be written as:

$$I_i = \sum_{(l,i,j)\in E} Y_{ij} V_j \tag{A.21}$$

Having the branch and net bus currents computed, the AC power flow equations can be formulated.

A.4 AC power flow equations

With the net bus currents computed by A.21, the net complex power injection at each bus $i \in N$ can be computed as:

$$S_i^n = P_i^n + jQ_i^n = V_i I_i^* \tag{A.22}$$

Substituting Equation A.21 into A.22 results in the following:

$$S_i^n = V_i \sum_{(l,i,j)\in E} Y_{ij}^* V_j^* \tag{A.23}$$

Rewrite the complex voltages and bus admittances as magnitude and phase angle form: $V_i = |V_i|\angle\delta_i$, $Y_{ij} = |Y_{ij}|\angle\theta_{ij}$, the Equation A.23 can be written as:

$$S_i^n = |V_i| \sum_{(l,i,j)\in E} |Y_{ij}||V_j|\angle(\delta_i - \theta_{ij} - \delta_j) \tag{A.24}$$

Given the complex voltages at all buses, the net real and reactive power injections at bus i can be computed by:

$$P_i^n = |V_i| \sum_{(l,i,j)\in E} |Y_{ij}||V_j| cos(\delta_i - \theta_{ij} - \delta_j) \tag{A.25}$$

$$Q_i^n = |V_i| \sum_{(l,i,j)\in E} |Y_{ij}||V_j| sin(\delta_i - \theta_{ij} - \delta_j) \tag{A.26}$$

At any bus i, the net complex power injections are the differences between the power generations and loads. This relation formulates the AC power flow equations:

$$S_i^g - S_i^d = |V_i| \sum_{(l,i,j)\in E} |Y_{ij}||V_j|\angle(\delta_i - \theta_{ij} - \delta_j) \tag{A.27}$$

With the complex bus voltages, the power flow at branch (l, i, j) can be computed by:

$$S_{ij} = V_i I_{ij}^* \tag{A.28}$$

$$S_{ji} = V_j I_{ji}^* \tag{A.29}$$

Substituting Equation A.9 into A.28 results in:

$$S_{ij} = \left({y_{ij}^s}^* - j\frac{b_{ij}^{sh}}{2} \right) \frac{1}{\tau_l^2} |V_i|^2 - \frac{{y_{ij}^s}^*}{\tau_l e^{j\theta_l^t}} V_i V_j^* \tag{A.30}$$

Similarly, substituting Equation A.11 into A.29 results in:

$$S_{ji} = \left({y_{ji}^s}^* - j\frac{b_{ji}^{sh}}{2} \right) \frac{1}{\tau_l^2} |V_j|^2 - \frac{{y_{ji}^s}^*}{\tau_l e^{-j\theta_l^t}} V_j V_i^* \tag{A.31}$$

At any bus i, the net complex power injection equals to the sum of all branch powers connected to this bus:

$$S_i^n = \sum_{(l,i,j) \in E} (S_{ij} + S_{ji}) \tag{A.32}$$

Since in the model of power systems, the loads S_i^d and the bus admittance matrix $\mathbf{Y}_{bus}$ are given and usually constant, the AC power flow equations A.27 can be expressed as functions of complex voltages V and complex power generations S_i^g:

$$g(V, S_i^g) = S_i^g - S_i^d - |V_i| \sum_{(l,i,j) \in E} |Y_{ij}||V_j| \angle(\delta_i - \theta_{ij} - \delta_j) = 0 \tag{A.33}$$

With the AC power flow equations, the AC-OPF problem is defined in section 3.3.

Bibliography

[AH17] Charles Audet and Warren Hare. *Derivative-free and blackbox optimization*. Vol. 2. Springer, 2017.

[AM17] Florian Augustin and Youssef M Marzouk. "A trust-region method for derivative-free nonlinear constrained stochastic optimization". In: *arXiv preprint arXiv:1703.04156* (2017).

[AV20] Mohammed Albadi and K Volkov. "Power flow analysis". In: *Computational Models in Engineering* (2020), pp. 67–88.

[Bab+19] Sogol Babaeinejadsarookolaee et al. "The power grid library for benchmarking ac optimal power flow algorithms". In: (2019).

[BB88] Jonathan Barzilai and Jonathan M Borwein. "Two-point step size gradient methods". In: *IMA journal of numerical analysis* 8.1 (1988), pp. 141–148.

[BD87] George EP Box and Norman R Draper. *Empirical model-building and response surfaces*. John Wiley & Sons, 1987.

[Ber+17] Felix Berkenkamp et al. "Safe model-based reinforcement learning with stability guarantees". In: *Advances in neural information processing systems* 30 (2017).

[BI96] Russell R Barton and John S Ivey Jr. "Nelder-Mead simplex modifications for simulation optimization". In: *Management Science* 42.7 (1996), pp. 954–973.

[Bib+20] Adel Bibi et al. "A stochastic derivative-free optimization method with importance sampling: Theory and learning to control". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 3275–3282.

[BKS21] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics". In: *Machine Learning* (2021), pp. 1–35.

[BS15] Felix Berkenkamp and Angela P Schoellig. "Safe and robust learning control with Gaussian processes". In: *2015 European Control Conference (ECC)*. IEEE. 2015, pp. 2496–2501.

[BSK16] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. "Safe controller optimization for quadrotors with Gaussian processes". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 491–496.

[BSV14] Afonso S Bandeira, Katya Scheinberg, and Luis Nunes Vicente. "Convergence of trust-region methods based on probabilistic models". In: *SIAM Journal on Optimization* 24.3 (2014), pp. 1238–1264.

[CDV08] AL Custódio, JE Dennis, and Luis Nunes Vicente. "Using simplex gradients of nonsmooth functions in direct search methods". In: *IMA journal of numerical analysis* 28.4 (2008), pp. 770–784.

[Cha12] Kuo-Hao Chang. "Stochastic Nelder–Mead simplex method–A new globally convergent direct search method for simulation optimization". In: *European journal of operational research* 220.3 (2012), pp. 684–694.

[Cha18] Spyros Chatzivasileiadis. "Optimization in modern power systems". In: *Lecture Notes. Tech. Univ. of Denmark. Available online: https://arxiv.org/pdf/1811.00943. pdf* (2018).

[CHV15] Carleton Coffrin, Hassan L Hijazi, and Pascal Van Hentenryck. "The QC relaxation: A theoretical and computational study on optimal power flow". In: *IEEE Transactions on Power Systems* 31.4 (2015), pp. 3008–3018.

[CHW13] Kuo-Hao Chang, L Jeff Hong, and Hong Wan. "Stochastic trust-region response-surface method (STRONG)—A new response-surface framework for simulation optimization". In: *INFORMS Journal on Computing* 25.2 (2013), pp. 230–243.

[CL13] Andrew R Conn and Sébastien Le Digabel. "Use of quadratic models with mesh-adaptive direct search for constrained black box optimization". In: *Optimization Methods and Software* 28.1 (2013), pp. 139–158.

[CMS18] Ruobing Chen, Matt Menickelly, and Katya Scheinberg. "Stochastic optimization using a trust-region method and random models". In: *Mathematical Programming* 169.2 (2018), pp. 447–487.

[CST97] Andrew R Conn, Katya Scheinberg, and Ph L Toint. "On the convergence of derivative-free methods for unconstrained optimization". In: *Approximation theory and optimization: tributes to MJD Powell* (1997), pp. 83–108.

[CSV09] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. "Global convergence of general derivative-free trust-region algorithms to first-and second-order critical points". In: *SIAM Journal on Optimization* 20.1 (2009), pp. 387–415.

[CT96] Andrew R Conn and Philippe L Toint. "An algorithm using quadratic interpolation for unconstrained derivative free optimization". In: *Nonlinear optimization and applications*. Springer, 1996, pp. 27–47.

[DF06] Geng Deng and Michael C Ferris. "Adaptation of the UOBYQA algorithm for noisy functions". In: *Proceedings of the 2006 winter simulation conference*. IEEE. 2006, pp. 312–319.

[DGG84] Renato De Leone, Manlio Gaudioso, and Luigi Grippo. "Stopping criteria for linesearch methods without derivatives". In: *Mathematical programming* 30.3 (1984), pp. 285–300.

[DKC13] Josip Djolonga, Andreas Krause, and Volkan Cevher. "High-dimensional gaussian process bandits". In: *Advances in neural information processing systems* 26 (2013).

[DMR08] MA Diniz-Ehrhardt, JM Martınez, and Marcos Raydán. "A derivative-free nonmonotone line-search technique for unconstrained optimization". In: *Journal of computational and applied mathematics* 219.2 (2008), pp. 383–397.

[DT97] JE Dennis and Virginia Torczon. "Managing approximation models in optimization". In: *Multidisciplinary design optimization: State-of-the-art* 5 (1997), pp. 330–347.

[Dui+17] Rikky RPR Duivenvoorden et al. "Constrained Bayesian optimization with particle swarms for safe adaptive controller tuning". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 11800–11807.

[Fan+11] Xi Fang et al. "Smart grid—The new and improved power grid: A survey". In: *IEEE communications surveys & tutorials* 14.4 (2011), pp. 944–980.

[Fra18] Peter I Frazier. "A tutorial on Bayesian optimization". In: *arXiv preprint arXiv:1807.02811* (2018).

[GLL88] L Grippo, F Lampariello, and S Lucidi. "Global convergence and stabilization of unconstrained minimization methods without derivatives". In: *Journal of Optimization Theory and Applications* 56.3 (1988), pp. 385–406.

[GR15] Luigi Grippo and Francesco Rinaldi. "A class of derivative-free nonmonotone optimization algorithms employing coordinate rotations and gradient approximations". In: *Computational Optimization and Applications* 60.1 (2015), pp. 1–33.

[GS07] Luigi Grippo and Marco Sciandrone. "Nonmonotone derivative-free methods for nonlinear equations". In: *Computational Optimization and Applications* 37.3 (2007), pp. 297–328.

[HJ61] Robert Hooke and T. A. Jeeves. "" Direct Search" Solution of Numerical and Statistical Problems". In: *J. ACM* 8.2 (Apr. 1961), pp. 212–229. ISSN: 0004-5411. DOI: 10.1145/321062.321069. URL: https://doi.org/10.1145/321062.321069.

[Ino+20] Masaki Inoue et al. "Optimal power flow design for enhancing dynamic performance: Potentials of reactive power". In: *IEEE Transactions on Smart Grid* 12.1 (2020), pp. 599–611.

[Jea71] Céa Jean. *Optimisation : théorie et algorithmes / Jean Céa, ...* fre. Méthodes mathématiques de l'informatique. Paris: Dunod, 1971.

[Kol+18] Torsten Koller et al. "Learning-based model predictive control for safe exploration". In: *2018 IEEE conference on decision and control (CDC)*. IEEE. 2018, pp. 6059–6066.

[KZ10] Sujin Kim and Dali Zhang. "Convergence properties of direct search methods for stochastic optimization". In: *Proceedings of the 2010 winter simulation conference*. IEEE. 2010, pp. 1003–1011.

[LB16] Jeffrey Larson and Stephen C Billups. "Stochastic derivative-free optimization using a trust region framework". In: *Computational Optimization and applications* 64.3 (2016), pp. 619–645.

[LMW19] Jeffrey Larson, Matt Menickelly, and Stefan M Wild. "Derivative-free optimization methods". In: *Acta Numerica* 28 (2019), pp. 287–404.

[LS02] Stefano Lucidi and Marco Sciandrone. "On the global convergence of derivative-free methods for unconstrained optimization". In: *SIAM Journal on Optimization* 13.1 (2002), pp. 97–116.

[MTK17] Mukund B Maskar, AR Thorat, and Iranna Korachgaon. "A review on optimal power flow problem and solution methodologies". In: *2017 International Conference on Data Management, Analytics and Innovation (ICDMAI)*. IEEE. 2017, pp. 64–70.

[MTZ78] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. "The application of Bayesian methods for seeking the extremum". In: *Towards global optimization* 2.117-129 (1978), p. 2.

[NM65] John A Nelder and Roger Mead. "A simplex method for function minimization". In: *The computer journal* 7.4 (1965), pp. 308–313.

[NS17] Yurii Nesterov and Vladimir Spokoiny. "Random gradient-free minimization of convex functions". In: *Foundations of Computational Mathematics* 17.2 (2017), pp. 527–566.

[Pas+18] Raghu Pasupathy et al. "On sampling rates in simulation-based recursions". In: *SIAM Journal on Optimization* 28.1 (2018), pp. 45–73.

[Pic+16] Victor Picheny et al. "Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian". In: *Advances in neural information processing systems* 29 (2016).

[PN21] Parikshit Pareek and Hung D Nguyen. "State-Aware Stochastic Optimal Power Flow". In: *Sustainability* 13.14 (2021), p. 7577.

[Pou16] Tony Pourmohamad. *Combining multivariate stochastic process models with filter methods for constrained optimization*. University of California, Santa Cruz, 2016.

[Pow03] Michael JD Powell. "On trust region methods for unconstrained minimization without derivatives". In: *Mathematical programming* 97.3 (2003), pp. 605–623.

[Pow04] MJD Powell. "On the use of quadratic models in unconstrained minimization without derivatives". In: *Optimization Methods and Software* 19.3-4 (2004), pp. 399–411.

[Pow06] Michael JD Powell. "The NEWUOA software for unconstrained optimization without derivatives". In: *Large-scale nonlinear optimization*. Springer, 2006, pp. 255–297.

[Pow10] MJD Powell. "On the convergence of a wide range of trust region methods for unconstrained optimization". In: *IMA journal of numerical analysis* 30.1 (2010), pp. 289–301.

[Pow64] Michael JD Powell. "An efficient method for finding the minimum of a function of several variables without calculating derivatives". In: *The computer journal* 7.2 (1964), pp. 155–162.

[Pow65] MJD Powell. "A method for minimizing a sum of squares of non-linear functions without calculating derivatives". In: *The Computer Journal* 7.4 (1965), pp. 303–307.

[Pow98] MJD Powell. "The use of band matrices for second derivative approximations in trust region algorithms". In: *Advances in Nonlinear Programming*. Springer, 1998, pp. 3–28.

[Pre+96] William H Press et al. *Numerical Recipes in Fortran 90: Numerical recipes in Fortran 77V. 2. Numerical recipes in Fortran 90*. Cambridge University Press, 1996.

[PS18] Courtney Paquette and Katya Scheinberg. "A stochastic line search method with convergence rate analysis". In: *arXiv preprint arXiv:1807.07994* (2018).

[PSZ89] Nitin R Patel, Robert L Smith, and Zelda B Zabinsky. "Pure adaptive search in Monte Carlo optimization". In: *Mathematical programming* 43.1 (1989), pp. 317–328.

[PT17] Margherita Porcelli and Philippe L Toint. "BFO, a trainable derivative-free brute force optimizer for nonlinear bound-constrained optimization and equilibrium computations with continuous and discrete variables". In: *ACM Transactions on Mathematical Software (TOMS)* 44.1 (2017), pp. 1–25.

[Red+16] Sashank J Reddi et al. "Stochastic variance reduction for nonconvex optimization". In: *International conference on machine learning*. PMLR. 2016, pp. 314–323.

[Rob52] Herbert Robbins. "Some aspects of the sequential design of experiments". In: *Bulletin of the American Mathematical Society* 58.5 (1952), pp. 527–535.

[RS13] Luis Miguel Rios and Nikolaos V Sahinidis. "Derivative-free optimization: a review of algorithms and comparison of software implementations". In: *Journal of Global Optimization* 56.3 (2013), pp. 1247–1293.

[Sha+15] Bobak Shahriari et al. "Taking the human out of the loop: A review of Bayesian optimization". In: *Proceedings of the IEEE* 104.1 (2015), pp. 148–175.

[SHH62] WGRFR Spendley, George R Hext, and Francis R Himsworth. "Sequential application of simplex designs in optimisation and evolutionary operation". In: *Technometrics* 4.4 (1962), pp. 441–461.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25 (2012).

[SM11] Soliman Abdel-Hady Soliman and Abdel-Aal Hassan Mantawy. *Modern optimization techniques with applications in electric power systems*. Springer Science & Business Media, 2011.

[SSK18] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. "A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions". In: *Journal of Mathematical Psychology* 85 (2018), pp. 1–16.

[Ste+82] William D Stevenson et al. *Elements of power system analysis*. Vol. 4. McGraw-Hill New York, 1982.

[Sui+15] Yanan Sui et al. "Safe exploration for optimization with Gaussian processes". In: *International conference on machine learning*. PMLR. 2015, pp. 997–1005.

[Sui+18] Yanan Sui et al. "Stagewise safe bayesian optimization with gaussian processes". In: *International conference on machine learning*. PMLR. 2018, pp. 4781–4789.

[Tay15] Joshua Adam Taylor. *Convex optimization of power systems*. Cambridge University Press, 2015.

[TBK16] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. "Safe exploration in finite markov decision processes with gaussian processes". In: *Advances in Neural Information Processing Systems* 29 (2016).

[UKK19] Ilnura Usmanova, Andreas Krause, and Maryam Kamgarpour. "Log barriers for safe non-convex black-box optimization". In: *arXiv preprint arXiv:1912.09478* (2019).

[VKB20] Floris Van Breugel, J Nathan Kutz, and Bingni W Brunton. "Numerical differentiation of noisy data: A unifying multi-objective optimization framework". In: *IEEE Access* 8 (2020), pp. 196865–196877.

[Win70] David Henry Winfield. "Function and functional optimization by interpolation in data tables". PhD thesis. Harvard University, 1970.

[Wri96] Margaret H Wright. "Direct search methods: Once scorned, now respectable". In: *Pitman Research Notes in Mathematics Series* (1996), pp. 191–208.

[WSI10] Xi-Fan Wang, Yonghua Song, and Malcolm Irving. *Modern power systems analysis*. Springer Science & Business Media, 2010.

[ZM20] Ray D. Zimmerman and Carlos E. Murillo-Sánchez. *MATPOWER*. Version 7.1. Oct. 2020. DOI: 10.5281/zenodo.4074135. URL: https://doi.org/10.5281/zenodo.4074135.

[ZMT11] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. "MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education". In: *IEEE Transactions on Power Systems* 26.1 (2011), pp. 12–19. DOI: 10.1109/TPWRS.2010.2051168.

[ZS92] Zelda B Zabinsky and Robert L Smith. "Pure adaptive search in global optimization". In: *Mathematical programming* 53.1 (1992), pp. 323–338.